UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Estudio e impletentación de la API de Whatsapp Bussines Cloud para una aplicación web

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Soto Navarro, Aaron

Tutor/a: López Patiño, José Enrique

CURSO ACADÉMICO: 2023/2024

Resumen

La API de WhatsApp Business Cloud ha surgido como una herramienta valiosa para integrar la plataforma de WhatsApp en aplicaciones web. El presente proyecto tiene como objetivo llevar a cabo un estudio detallado de esta API, para posteriormente llevar a cabo su implementación en una aplicación web utilizando el framework de PHP Laravel.

La finalidad principal es demostrar las principales funcionalidades que ofrece y gestionar toda la información relevante para la comunicación entre empresas y clientes. En este estudio se analizará la documentación proporcionada por la empresa Meta y en base a ella, como ya se ha comentado, se llevará a cabo una implementación concreta en una aplicación web que muestre cómo operan las diferentes funcionalidades que incorpora.

Con este enfoque, se podrá obtener una visión completa del producto, evaluando tanto sus ventajas como limitaciones a la hora de su integración, así como obtener un conocimiento más profundo sobre las posibilidades de esta herramienta y su capacidad para mejorar la comunicación entre empresas y clientes.

Resum

La API de WhatsApp Business Cloud ha sorgit com una eina valuosa per integrar la plataforma de WhatsApp en aplicacions web. El present projecte té com a objectiu dur a terme un estudi detallat d'aquesta API, per posteriorment dur a terme la seva implementació en una aplicació web utilitzant el framework de PHP Laravel.

La finalitat principal és demostrar les principals funcionalitats que ofereix i gestionar tota la informació rellevant per a la comunicació entre empreses i clients. En aquest estudi s'analitzarà la documentació proporcionada per l'empresa Meta i en base a ella, com ja s'ha comentat, es durà a terme una implementació concreta en una aplicació web que mostri com operen les diferents funcionalitats que incorpora.

Amb aquest enfocament, es podrà obtenir una visió completa del producte, avaluant tant les seves avantatges com limitacions a l'hora de la seva integració, així com obtenir un coneixement més profund sobre les possibilitats d'aquesta eina i la seva capacitat per millorar la comunicació entre empreses i clients.

Abstract

The WhatsApp Business Cloud API has emerged as a valuable tool for integrating the WhatsApp platform into web applications. This project aims to conduct a detailed study of this API, followed by its implementation in a web application using the PHP Laravel framework.

The main purpose is to demonstrate the key functionalities it offers and manage all the relevant information for communication between businesses and clients. This study will analyze the documentation provided by Meta, and based on it, as already mentioned, a specific implementation will be carried out in a web application that shows how the different functionalities it incorporates operate.

With this approach, a complete view of the product can be obtained, evaluating both its advantages and limitations at the time of integration, as well as gaining a deeper understanding of the possibilities of this tool and its ability to improve communication between businesses and clients.

Índice

Índice de figuras

Índice de tablas

Siglas y Términos

API	Aplication Programming Interface.
PHP	Hypertext Preprocessor.
JSON	JavaScript Object Notation.
EndPoint	punto final de una comunicación en una red para acceder desde una URL.
CLI	Command Line Interface.
CRUD	Create, Read, Update y Delete **.**
ORM	Object-Relational Mapping.
SQL	Structured Query Language.
HTTP	Hypertext Transfer Protocol.
CSS	Cascading Style Sheets.
AJAX	Asynchronous JavaScript And XML
URL	Uniform Resource Identifier

Capítulo 1. Introducción

1.1 Contexto

Este proyecto tiene como objetivo mejorar los servicios de atención que las empresas ofrecen a sus clientes. WhatsApp es una de las aplicaciones de mensajería instantánea más utilizadas, por lo que incorporar esta herramienta en los equipos de soporte resulta altamente interesante para proporcionar un servicio eficaz y rápido, ya que ofrece una vía alternativa a la comunicación telefónica convencional.

Sin embargo, la integración de WhatsApp como forma de comunicación supuso, en la mayoría de los casos, la utilización por parte de los trabajadores de sus propios números personales o directos de empresa, así como el uso de sus propios terminales, lo que aumentaba el número de dispositivos a utilizar y suponía, además, problemas de privacidad.

Por ello, en 2015, WhatsApp ofreció la posibilidad de utilizar su servicio a través de una plataforma web, y en 2018 ofreció un servicio ampliado con funcionalidades orientadas al mundo de la pequeña empresa, denominado WhatsApp Business. No obstante, ambas versiones de WhatsApp seguían orientadas a que un mismo número telefónico pudiera ser usado por un único usuario, lo que dificultaba su uso en entornos con varios usuarios (agentes de servicio) operando en simultáneo (multiagente).

Además, eran herramientas independientes, que debían ser utilizadas adicionalmente a las propias herramientas de gestión que ya venían utilizando los agentes, lo que dificultaba su uso. Por ello, Meta proporcionó una API que permitiera cubrir la necesidad de las empresas de comunicarse con sus clientes a través de WhatsApp de una manera escalable, brindando la capacidad de atender cientos o miles de mensajes diariamente con varios agentes.

Esta API se denomina Business Cloud API, y no está concebida como una herramienta independiente, sino como una interfaz de aplicación que permite desarrollar programas personalizados o integrarla en programas de gestión ya existentes según las necesidades de cada empresa, facilitando así el trabajo diario de los agentes, que podrían utilizar WhatsApp como una funcionalidad más dentro de su herramienta de atención a los clientes.

1.2 Objetivo

El objetivo central de este Trabajo de Fin de Grado es llevar a cabo la implementación de una aplicación web mediante el framework de PHP Laravel, con el propósito de lograr una estrecha integración con la API proporcionada por la empresa Meta. Esta API se presenta como una valiosa opción para aquellas empresas que desean incorporar la funcionalidad de WhatsApp en sus plataformas web sin depender de intermediarios.

El enfoque principal de la integración entre Laravel y la API de WhatsApp radica en desarrollar un código simplificado capaz de simular una conversación entre un cliente y un usuario, permitiendo así el seguimiento fluido de las interacciones.

Para alcanzar este propósito, se llevará a cabo el desarrollo del código primero en local de forma que se pueda ir desarrollando cómodamente, tras esto se almacenara en una plataforma en la nube previamente configurada, lo cual facilitara el acceso a través de internet gracias a un dominio y para subir este código a dicha plataforma se ha adoptado GitHub como repositorio para la gestión de las diferentes versiones del código.

Con el presente proyecto, se busca demostrar la viabilidad y utilidad de esta implementación en términos de optimización de la experiencia del cliente y la adopción de un canal de comunicación efectivo entre cliente y empresa.

1.3 Metodología

Para la realización del TFG se ha utilizado una metodología por objetivos, donde para poder pasar al siguiente objetivo se ha tenido que realizar de forma eficaz el anterior paso con la finalidad de no encontrar errores futuros, con las verificaciones necesarias en los puntos de implementación.

1.3.1 Distribución de tareas

- Tarea 1: Selección del Tema de Investigación y Desarrollo.
- Tarea 2: Adquisición de Competencias en el Lenguaje de Programación Escogido.
- Tarea 3: Configuración del Entorno de Desarrollo y Despliegue.
- Tarea 4: Análisis y Estudio Exhaustivo de la Documentación de la API.
- Tarea 5: Diseño Detallado de la Estructura de la Base de Datos.
- Tarea 6: Creación del Diseño de la Interfaz de Usuario.
- Tarea 7: Implementación de la Lógica para el Envío y Recepción de Mensajes.
- Tarea 8: Investigación y Comprensión del Funcionamiento de un Webhook.
- Tarea 9: Integración de Mecanismos para el Envío y Recepción de Mensajes.
- Tarea 10: Actualización de la Interfaz de Usuario para Visualización en Tiempo Real.
- Tarea 11: Ejecución de Pruebas Rigurosas y Verificaciones Exhaustivas.
- Tarea 12: Optimización de Rendimiento y Corrección de Errores Identificados.

Tarea	Fecha inicio	Fecha Fin
Tarea 1	27/07/2022	01/08/2022
Tarea 2	02/08/2022	10/08/2022
Tarea 3	18/08/2022	22/08/2022
Tarea 4	23/08/2022	15/09/2022
Tarea 5	16/09/2022	19/09/2022
Tarea 6	20/09/2022	24/09/2022
Tarea 7	25/09/2022	28/09/2022
Tarea 8	29/10/2022	10/10/2022
Tarea 9	11/10/2022	17/10/2022
Tarea 10	18/10/2022	02/11/2022
Tarea 11	03/11/2022	10/11/2022
Tarea 12	01/11/2023	15/11/2023

Tabla 1 Organización de tareas

1.4 Estructura de la memoria

La memoria se divide en 7 capítulos. A continuación, se presentará una breve descripción que resume el contenido abordado en cada uno de estos capítulos.

- Capítulo 1: En este capítulo se introduce el proyecto mediante una introducción, donde se presentan y delinean las metas, detallando los objetivos.
- Capítulo 2: Aquí se describe cómo utilizar la plataforma de WhatsApp Business y se presentan sus funcionalidades.
- Capítulo 3: En este capítulo se presentarán y explicarán las tecnologías y herramientas utilizadas a lo largo del proyecto.
- Capítulo 4: Se muestra la implementación de la aplicación web en este capítulo, junto con las configuraciones necesarias para cada tipo de función.
- Capítulo 5: En este capítulo se exponen las funcionalidades de la aplicación y se explica cómo hacer uso de su interfaz gráfica.
- Capítulo 6: Aquí se hace una conclusión del trabajo realizado y propuestas para futuro.
- Capítulo 7: Bibliografía

Capítulo 2. Plataforma de WhatsApp Bussines

La Plataforma de WhatsApp Business facilita la configuración de aplicaciones que buscan integrar los productos ofrecidos por Meta. Esta plataforma permite, de manera gráfica e intuitiva, llevar a cabo los primeros pasos en la configuración de dichos productos, simplificando el proceso para las empresas, proporcionando herramientas para la configuración de diversos elementos esenciales, garantizando el uso y correcto funcionamiento de la aplicación. Dado que la eficiencia de esta herramienta depende de ajustes específicos, es crucial adaptar y personalizar diferentes parámetros para asegurar una implementación óptima de la API.

Esta plataforma también provee información crucial para el buen funcionamiento de las aplicaciones web. Destaca, por ejemplo, la capacidad de notificar diferentes eventos que ocurren tanto internamente como en acciones vinculadas a la API de mensajes, todo ello relacionado con la gestión y administración de los activos.

2.1 Condiciones de uso

Para acceder y utilizar la plataforma, es esencial comprender y adherirse a las condiciones y restricciones dictadas por Meta. Estas directrices son cruciales cuando las empresas incorporan el sistema de mensajería instantánea de WhatsApp con objetivos comerciales. Cumplir con estas normas garantiza una experiencia óptima y previene malentendidos o mal uso de la herramienta.

La solución API de WhatsApp Business tiene requisitos específicos, entre los cuales destaca la necesidad de obtener un consentimiento explícito de los destinatarios antes de enviar mensajes por WhatsApp. Debe quedar claro que están de acuerdo en recibir tales comunicaciones y se debe especificar el nombre de la entidad emisora. Es vital que las empresas asuman esta responsabilidad.

El inicio de una conversación con un cliente puede llevarse a cabo mediante Plantillas de Mensajes Aprobadas. Estas, sin embargo, requieren una revisión previa y deben utilizarse únicamente para el propósito detallado en el formulario de revisión.

Una vez establecida una conversación, existe una ventana de 24 horas para el intercambio fluido de mensajes. Pasado este tiempo, y si la empresa desea continuar la comunicación, debe hacerlo mediante una plantilla aprobada.

Si se recurre a respuestas automáticas durante este intervalo, es crucial ofrecer opciones alternativas de comunicación, como:

- Chat en vivo con un representante
- Número de teléfono
- Correo electrónico
- Asistencia web
- Visitas presenciales (ej. tienda o sucursal bancaria)
- Formulario de contacto en línea

Las empresas deben manejar esta herramienta de comunicación con prudencia, garantizando la confidencialidad y seguridad de la información del cliente. Es imperativo evitar solicitar o compartir datos sensibles, como detalles bancarios o identificaciones personales. [1]

2.2 Acceso y uso de la plataforma

Una vez visto las condiciones que nos pone Meta para usar la API pasamos a ver como comenzar a usar la plataforma y como empezar a desarrollar la aplicación a partir de los recursos que se ofrecen, desde esta plataforma es desde donde se van a realizar numerosas configuraciones y administrar activos necesarios para poder operar con esta API.

Para iniciar la utilización de la API de la nube de WhatsApp Business, es necesario crear una cuenta de desarrollo en Meta, así como generar una aplicación para desarrolladores en Meta. En consecuencia, el primer paso consiste en registrarse como desarrollador en Meta. Este proceso requiere disponer de una cuenta de Facebook, a través de la cual se puede realizar el registro en Meta for Developers, hay que pasar una serie de pasos para la autenticación con la finalidad de que la aplicación que se va a crear tenga una capa de seguridad extra.

Una vez registrado como desarrollador, ya se puede acceder al panel de aplicaciones de Meta, desde aquí se pueden ver las diferentes aplicaciones creadas, Meta ofrece diferentes productos con los que realizar integraciones, por lo que desde el panel que se ve en la Figura 1 se puede gestionar las diferentes aplicaciones creadas, en este caso únicamente hay una aplicación dedicada a la API de WhatsApp.

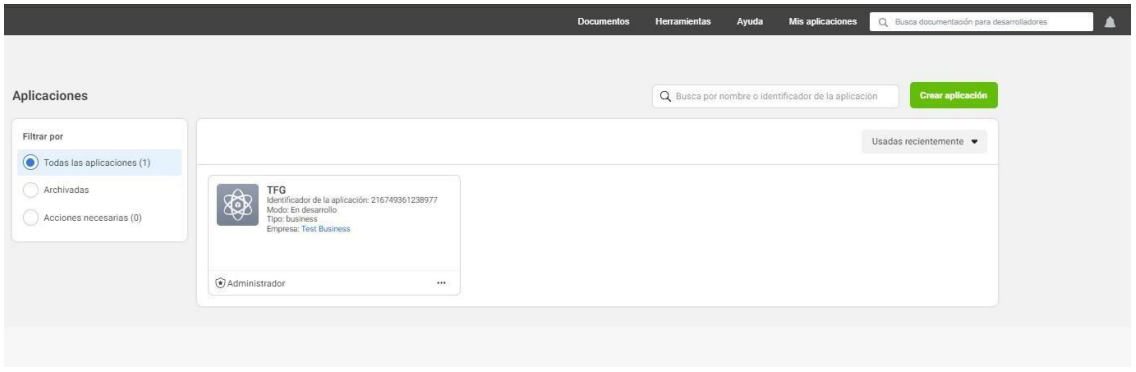


Figura 1 Panel de aplicaciones plataforma WhatsApp Bussines

Para comenzar, al presionar el botón de "Crear Aplicación", se abrirá una página que muestra diversos casos de uso. Dependiendo del tipo de aplicación que se esté desarrollando, será necesario elegir el caso correspondiente. Como se observa en la Figura 2, el primer caso es una práctica común en muchas aplicaciones, que implica la incorporación del inicio de sesión a través de Facebook

En el contexto de aplicaciones de este tipo, se implementa una redirección a la aplicación de Facebook para el proceso de registro. Al aceptar los términos, los usuarios pueden crear una cuenta en la aplicación deseada de manera rápida y sencilla.

El segundo caso de uso es similar al anterior, pero se aplica específicamente a aplicaciones de juegos. Esto simplifica el proceso de registro y mejora la comodidad del usuario. Sin embargo, en el caso de esta aplicación en particular, se opta por prescindir de esta funcionalidad.

En consecuencia, se elige la opción denominada "Otro", que permitirá continuar con la configuración de la aplicación de acuerdo con las necesidades y requerimientos específicos que se presenten.

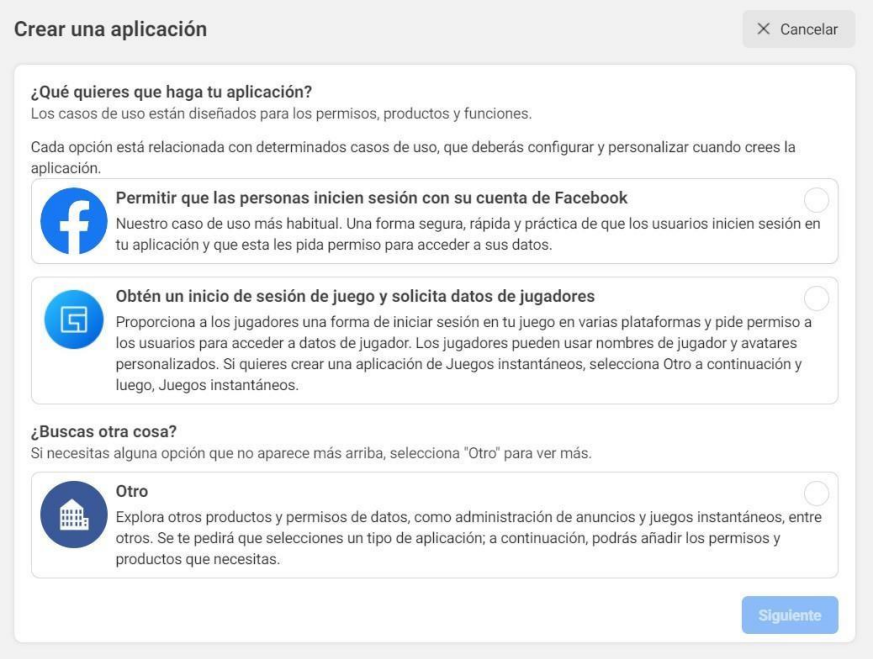


Figura 2 Casos de uso para aplicaciones plataforma WhatsApp Bussines

Después de elegir esta opción, se despliegan varias alternativas que exploran el tipo de aplicación a desarrollar. Para acceder a la API de WhatsApp, es crucial seleccionar la pestaña "Empresa", ya que esta elección otorga acceso a una gama diversa de activos comerciales, permisos administrativos y productos empresariales, tal como se muestra en la Figura 3, que ilustra esta información.

Finalmente, para concluir la configuración, se debe asignar un nombre a la aplicación, proporcionar una dirección de correo electrónico y, en caso de disponer previamente de una cuenta empresarial, hacer mención de ella. Sin embargo, es posible continuar sin ella y crearla posteriormente.

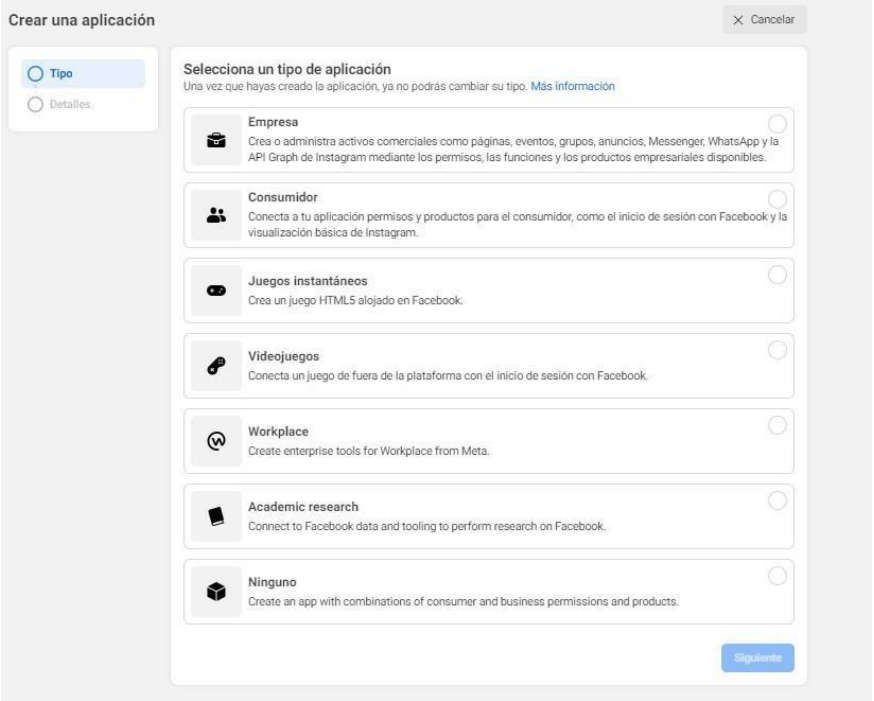


Figura 3 Selector tipo de aplicación plataforma WhatsApp Bussines

Una vez la aplicación ha sido creada, se puede acceder al panel de control. En este panel, se llevarán a cabo diversas configuraciones necesarias para operar con la API de WhatsApp, tal como se ilustra en la Figura 4. Desde esta interfaz, se podrán observar varios eventos y se podrá agregar los productos indispensables para la creación de la aplicación.

En el panel de control se incluyen los productos "WhatsApp" y "Webhook", los cuales son esenciales para el uso de esta API. Estos dos productos son necesarios y fundamentales en el proceso.

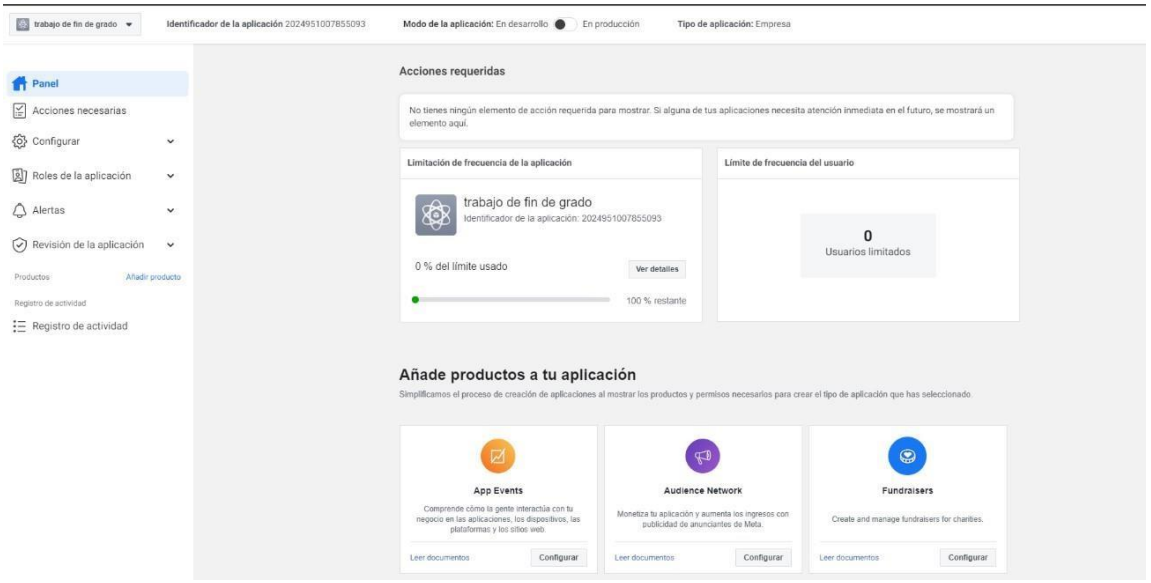


Figura 4 Panel de control plataforma WhatsApp Bussines

2.2.1 Producto WhatsApp

El producto de WhatsApp constituye una herramienta que facilita la realización de variadas configuraciones vinculadas a la aplicación, y posibilita la interacción con la API de una manera más visual. Ofrece información acerca de los diversos EndPoints disponibles para el envío de mensajes.

Mediante este producto, se puede llevar a cabo una configuración inicial de la aplicación, efectuar pasos para realizar pruebas iniciales con la API, agregar números de teléfono, métodos de pago y otras configuraciones. Además, posibilita la apertura de tickets para informar sobre incidencias relacionadas con el uso de la aplicación.

La configuración se presenta como en la Figura 5. A partir de este punto, es posible iniciar la creación de la cuenta empresarial, donde se proporciona información de la cuenta. Desde aquí, se será redirigido a una página que permitirá la administración de activos, como plantillas o teléfonos. Estos activos resultan indispensables para el uso de esta API y serán abordados en detalle más adelante.

Además, dentro de este producto, se presenta la pestaña de configuración de la API. Aquí se proporciona un breve tutorial en el que, en primer lugar, se generará un token temporal válido por 24 horas, únicamente para fines de prueba. Esta etapa es donde se ilustran de manera simple los procesos de envío y recepción de mensajes. Esto permite tener una comprensión preliminar de cómo funcionan estas operaciones.

Estos pasos constituyen los requisitos para utilizar el producto de WhatsApp, que luego se implementarán en el código de la aplicación.

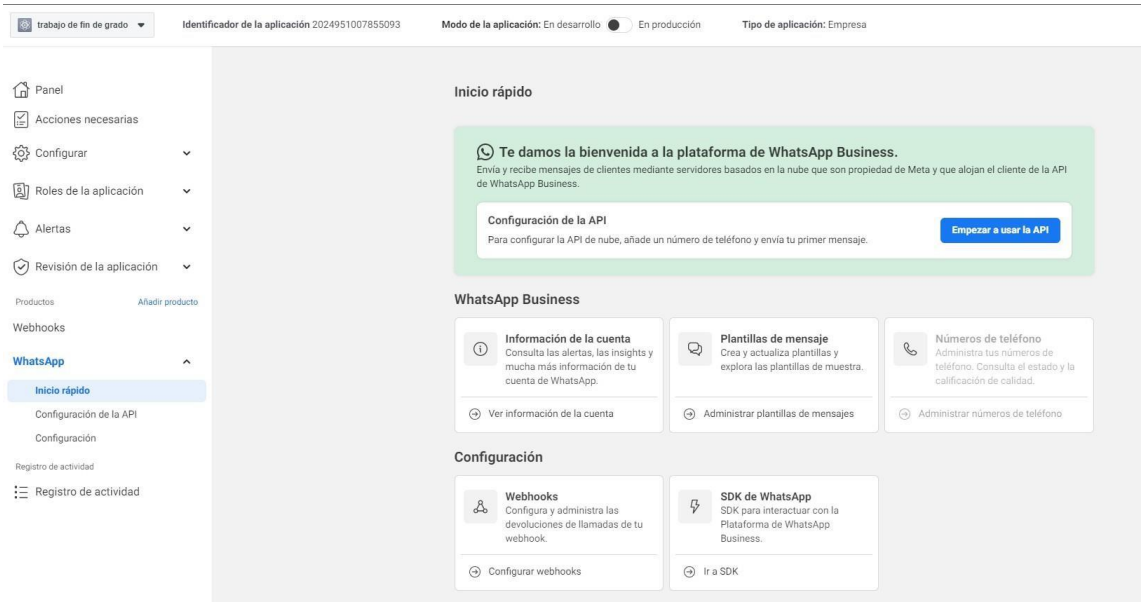


Figura 5 Panel de control producto WhatsApp

2.2.2 Producto Webhook

El producto Webhooks es una vía por la cual Meta se comunica con la aplicación a través de solicitudes HTTP en tiempo real. Esto habilita la recepción de información proveniente de diversos elementos que la aplicación ofrece. En el contexto de la API de WhatsApp Business, los Webhooks proveen a la aplicación actualizaciones y verificaciones de eventos que tienen lugar en la misma.

Estas actualizaciones incluyen la recepción de mensajes cuando un cliente responde, cambios en el estado del mensaje de "enviado" a "leído" o cuando surgen problemas con un tipo de mensaje específico. Para lograr esto, la plataforma ofrece la opción de suscribirse a varios elementos que enviarán estos eventos a la aplicación.

Adicionalmente, los Webhooks también posibilitan la recepción de información acerca de ciertos eventos que suceden en la plataforma de WhatsApp Business y que no están contemplados en la API. Esto abarca actualizaciones en el estado de las plantillas o los números de teléfono utilizados en la aplicación.

2.3 Plantillas

Las plantillas de WhatsApp son mensajes predefinidos utilizados para lograr una comunicación estructurada y automatizada a través de la plataforma de WhatsApp Business. Estas plantillas permiten enviar mensajes consistentes y personalizados a los usuarios, ya sea para notificaciones, confirmaciones de pedidos, recordatorios o actualizaciones.

Cada plantilla sigue un formato específico que se divide en dos partes: una estructura que define el diseño del mensaje e incluye campos variables personalizables, como el nombre, número de pedido o fecha, y contenido estático que contiene el texto fijo del mensaje que será enviado a todos los usuarios, independientemente de la información personalizada.

Las plantillas pasan por diferentes estados en su ciclo de vida. Para usar una plantilla personalizada, debe ser revisada y aprobada por Meta. Estos son los estados posibles de una plantilla:

- **Revisión:** En proceso de comprobación.
- **Rechazada:** Denegada por alguna razón.
- **Activa:** Aprobada para su uso.
- **En Pausa:** Temporalmente inactiva debido a valoraciones de clientes.
- **Desactivada:** Temporalmente inactiva debido a valoraciones.
- **Apelación solicitada:** Se ha recurrido al estado de la plantilla.

El estado de las plantillas no es definitivo, se revisan continuamente según criterios de clasificación. Las plantillas se califican en función de su uso y los comentarios de los clientes, lo cual afecta su calificación de calidad en el panel de control.

El ciclo de vida de una plantilla en WhatsApp Business inicia con su primer envío. Después de un período de tiempo determinado, esta entra en un proceso de revisión. Si la plantilla es aprobada durante este proceso, entonces se activa y se le espera una calificación de calidad.

Una vez activa, la plantilla es susceptible a los comentarios y calificaciones de los usuarios. Si recibe comentarios negativos de manera constante, su estado puede cambiar. En respuesta a calificaciones negativas, la plantilla puede pausarse. La primera vez se pausará por 3 horas y si vuelve a recibir calificaciones negativas, se pausará por 6 horas. Si esta situación se repite tres veces, la plantilla será desactivada.

Si en algún momento la plantilla es rechazada, existe la opción de presentar una apelación. Esta apelación debe contener una muestra de la plantilla y una descripción de sus usos previstos. Además, es importante destacar que si se decide editar la plantilla, esta deberá pasar nuevamente por el proceso de revisión antes de poder ser usada de nuevo [2].

Para poder llevar un seguimiento sobre los cambios de estados de la plantilla, se puede realizar una suscripción mediante Webhook a un campo especifico que permite recibir actualizaciones sobre los diferentes estados y procesos de revisión.

2.4 Números de teléfono

Para utilizar una cuenta de WhatsApp Business, es necesario contar con un número de teléfono válido destinado a ser empleado en la plataforma de WhatsApp Business. Para hacer uso de dicho número, deben satisfacerse los siguientes requisitos:

- El número debe pertenecer al propietario.
- Debe tener un código de país y de área, al igual que los números de teléfono fijos y móviles.
- Debe tener la capacidad de recibir llamadas de voz o mensajes SMS.
- No puede ser un código corto.
- No debe haber sido previamente utilizado en la plataforma de WhatsApp Business.

Si ya se dispone de otra cuenta que incluye un número de teléfono que se desea transferir a una nueva cuenta, se ofrece un proceso de migración para usarlo en la nueva aplicación. No obstante, es crucial recordar que un número de teléfono solo debe ser empleado en una cuenta de WhatsApp Business.

Cada empresa, ya sea verificada o no, tiene la opción de registrar múltiples números de teléfono. En el caso de empresas verificadas, se pueden registrar hasta 20 números, mientras que para las no verificadas, el límite es de 2 números. Esto conlleva implicaciones en las limitaciones de la plataforma.

Para considerar una empresa verificada es necesario pasar por un proceso de validación por parte de Meta, para asegurarse que los clientes están interactuando con una entidad legitima y reconocida.

Además, al crear la aplicación desde la plataforma de WhatsApp Business, se brinda un número de prueba que es apto para llevar a cabo pruebas y evaluar algunas funcionalidades. No obstante, es importante evitar utilizar este número de prueba en un entorno de producción en el que se empleen plantillas de ventas o marketing. [3]

2.5 Límites de mensajes

Los límites de mensajes se refieren al máximo de conversaciones que una empresa puede iniciar con cada número de teléfono en un periodo de 24 horas. La conversación comienza cuando la empresa envía una plantilla y finaliza pasadas otras 24 horas.

Al igual que con los números de teléfono, como se mencionó previamente, los límites varían según si la empresa está verificada o no. En el caso de no estar verificada, el límite es de 250 conversaciones iniciadas. En cambio, una empresa verificada puede tener un número ilimitado de clientes únicos a partir de las 1000 conversaciones, dependiendo de la calidad del número y la frecuencia de inicio de conversación.

La calidad se califica considerando cómo los destinatarios han recibido los mensajes en los últimos siete días, y se pondera en función de la antigüedad. Esta calificación se determina mediante una combinación de señales de calidad en las conversaciones entre empresas y usuarios. Ejemplos de estas señales incluyen comentarios de los usuarios, como bloqueos e informes, junto con los motivos que los usuarios proporcionan al bloquear a una empresa. Estas señales se utilizan para evaluar y asignar una calificación de calidad a la empresa en base a su interacción con los usuarios. [4]

2.6 Costes

Los precios de esta plataforma se basan en conversaciones, no en mensajes individuales. Estos precios pueden variar según el tipo de conversación que esté en curso, ya que cada conversación tiene una categoría específica.

Las conversaciones se dividen en categorías según la plantilla utilizada. Para iniciar una conversación de marketing, utilidad o autenticación, es necesario emplear una plantilla correspondiente. Dependiendo del tipo de plantilla, se iniciará una conversación acorde.

Cuando es el cliente quien inicia la conversación, comienza un período de atención al cliente. Al responder, se inicia la conversación de servicio. Si un cliente envía un mensaje, se establece un período de 24 horas, denominado "período de atención al cliente", durante el cual es posible responder con un mensaje libre. Si transcurren más de 24 horas, es necesario reenviar el mensaje usando una plantilla.

En situaciones donde ya existe una conversación en marcha con un cliente, como en el caso de una conversación de marketing, al enviar una plantilla de utilidad, se generarán dos conversaciones simultáneas bajo la misma conversación, y se cobrarán como dos conversaciones distintas.

Cada cuenta de WhatsApp Business recibe 1000 conversaciones gratuitas al mes. Sin embargo, estas no aplican a conversaciones de marketing, utilidad o autenticación. También existen las "conversaciones gratuitas desde el punto de acceso", en las que un cliente inicia una conversación a través de un anuncio de clic de WhatsApp o un botón de llamada de acción en una página de Facebook. Estas conversaciones, que pueden responderse con plantillas o mensajes libres, permanecen abiertas durante 72 horas y no tienen costo, pero se paga por el anuncio.

Las tarifas varían según la categoría de la conversación y la ubicación del cliente, utilizando los prefijos de llamada. Por ejemplo, en España, los precios para marketing son de 0.0509€, utilidad de 0.0315€, autenticación de 0.0283€ y servicio de 0.0305€.

Cuando se recibe un mensaje de cambio de estado en el Webhook, incluye un campo que proporciona información sobre el tipo de conversación al que pertenece, si es una conversación que se cobra y quién la inició. [5]

2.7 Comparativa local con la nube

La plataforma de WhatsApp Business ofrece dos alternativas para el uso de la API de mensajería a través de WhatsApp: una instalada en local y la otra en la nube.

Principales diferencias	API Local	API en la nube
Alojamiento	Hay que alojarla en los propios servidores de la empresa y en centros de datos	Almacenado por Meta
Mantenimiento	Hay que realizar actualizaciones de forma periódica del software de la API	Meta realiza las actualizaciones de forma automática.
Costes	Pago por el alojamiento en los servidores y por los mensajes enviados y recibidos según las conversaciones	Únicamente se paga por las conversaciones
Rendimiento	Máximo de 70 mensajes de texto por segundo en caso de conexiones únicas. Máximo de 250 mensajes de texto en conexiones múltiples. Mensajes multimedia reducen estos valores	Hasta 1000 mensajes por segundo, que pueden verse mermados por cargas elevadas del sistema que ofrece Meta.
Contenido multimedia	API de proveedores de contenidos multimedia disponibles	Para subir contenido multimedia hay que seguir una serie de pasos extra

Tabla 2 Diferencias entre la API de WhatsApp Bussines en la nube y en local

Como se muestra en la tabla, dependiendo del uso que se le vaya a dar a la API, puede ser beneficioso un sistema de alojamiento u otro. Para un caso como el de este trabajo, lo más ideal ha sido la implementación mediante la API en la nube, ya que se ahorran costes en infraestructura. Asimismo, para empresas que no dispongan de un gran equipo que pueda mantener actualizadas y en constante mantenimiento los servidores, la opción en la nube es más simple de manejar. En cambio, si se dispone de esos recursos, la API en local puede ser una mejor opción porque dotas a la aplicación de más funcionalidades propias sin depender tanto de la plataforma de WhatsApp Business, la cual puede que en situaciones donde la carga sea más excesiva, la aplicación se vea limitada por esta carga. [6]

Por la naturaleza de este proyecto la API en la nube se considera más adecuada, ya que puedes prescindir de el alojamiento en servidores propios que son costoso y además que únicamente se pagan en el caso que corresponda por lo mensajes enviados.

Capítulo 3. Tecnologías y Herramientas

3.1 tecnologías

3.1.1 Laravel

Laravel es un framework de PHP diseñado para el desarrollo de aplicaciones y servicios web. Este framework de código abierto facilita la creación de código gracias a su estructura, que proporciona un amplio abanico de funcionalidades. Se ha elegido este framework debido a su enfoque más estructurado en comparación con el PHP puro y a cierta experiencia previa en su uso. [7]

En este proyecto, se ha aprovechado su arquitectura Modelo-Vista-Controlador (MVC), que separa los datos de la aplicación, la lógica de control y la interfaz de usuario. Aunque Laravel ofrece una gama más amplia de recursos que se basan en este modelo, para proyectos de este tamaño, junto con algunos elementos adicionales, resulta adecuado, este tipo de modelo se muestra un funcionamiento básico en la Figura 6 de forma visual mostrando la comunicación entre los principales componentes de Laravel.

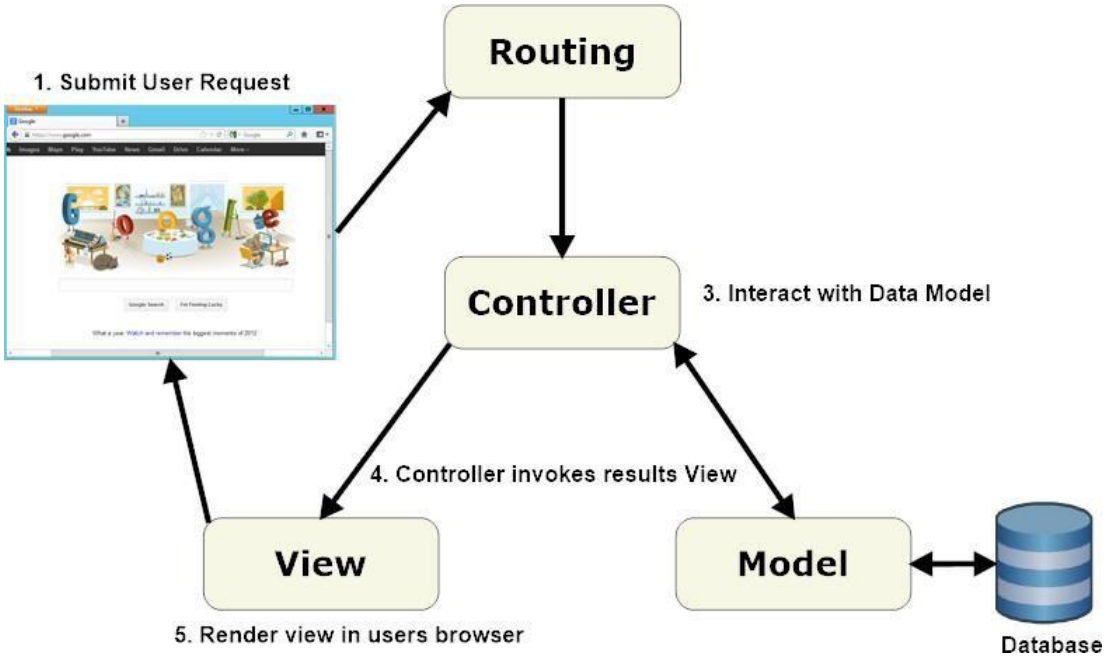


Figura 6 Arquitectura MVC de Laravel [8]

Una vez visto esto y con ya un entorno de desarrollo en Laravel desplegado se va a pasar a hablar de los diferentes componentes y como configurarlos durante el proceso de desarrollo de la aplicación.

El primer archivo que se configura es el ".env". En este archivo se asignan variables de entorno para el entorno de ejecución del código. Primero se establecen las variables de configuración de la aplicación, luego las credenciales para la conexión a la base de datos y otras contraseñas sensibles. Es crucial no subir este archivo a los repositorios, ya que la configuración varía según el entorno y contiene información delicada. El archivo se encuentra en el directorio raíz.

Posteriormente, se crean los componentes que interactúan con la base de datos, conocidos como Modelos en Laravel. Estos modelos representan los datos almacenados y proporcionan métodos para operaciones CRUD. Son generados automáticamente en el directorio "\app".

Los modelos se emplean para mejorar la interacción con la base de datos, reemplazando las consultas SQL estándar con el modelo Eloquent ORM. Esto simplifica la creación, lectura, manipulación y eliminación de registros.

Los controladores son esenciales en Laravel, ya que gestionan la lógica de la aplicación y las solicitudes HTTP. En este componente, se definen métodos que corresponden a acciones como mostrar vistas, almacenar en la base de datos o recibir datos JSON para actuar en consecuencia ante eventos. Los controladores se encuentran en el directorio "\app\Http\Controllers".

En este proyecto, los controladores son la pieza central, definiendo la lógica de la aplicación tanto en términos funcionales como en la manipulación de la base de datos a través de los modelos, así como el procesamiento de las solicitudes HTTP provenientes de distintas partes de la aplicación.

Las vistas son archivos que combinan código HTML y PHP para mostrar la interfaz de usuario de la aplicación web. Se ubican en el directorio "\resources\views". Para el código PHP en las vistas, se utiliza el motor de plantillas Blade, que permite estructuras de control, bucles y generación de código dinámico.

Las rutas son componentes que definen el manejo de las solicitudes HTTP entrantes. Se definen utilizando métodos del objeto "Route" predefinido en Laravel, accesible mediante la función "Route", o métodos específicos según el tipo de solicitud HTTP:GET, POST ,PUT ,PATCH , DELETE, entre otros. Aquí se definen las peticiones necesarias para el funcionamiento de la aplicación y la recepción de mensajes de WhatsApp.

Otro componente valioso son las migraciones, que permiten definir y modificar el esquema de la base de datos de manera programática. En lugar de modificar directamente la base de datos, las migraciones te permiten usar código PHP para crear, modificar o eliminar tablas, columnas, índices y otros elementos estructurales de la base de datos. Esto facilita la actualización de la base de datos durante el proceso de desarrollo.

Artisan es el CLI de Laravel. Es una herramienta versátil que permite realizar diversas tareas de desarrollo, administración y mantenimiento de aplicaciones Laravel. Con Artisan, es posible ejecutar comandos para generar código, migrar bases de datos, ejecutar pruebas y tareas programadas, entre otros. Ha sido extremadamente útil para trabajar con Laravel, ya que facilita la creación de componentes con una estructura base.

Estas son las principales herramientas que Laravel ofrece para el desarrollo de esta aplicación web, permitiendo un trabajo efectivo y organizado. Aunque Laravel ofrece más elementos, aquí se han enfocado en los más importantes para este proyecto.

3.1.2 Bootstrap

Bootstrap es un framework CSS diseñado para la creación de interfaces de usuario en el desarrollo de aplicaciones web. Proporciona plantillas de diseño y componentes de interfaz de usuario que facilitan la construcción de aplicaciones responsivas, adaptándose a diversos tamaños de pantalla y dispositivos. [9]

En este proyecto, se ha utilizado Bootstrap para diseñar la interfaz de usuario, otorgando color, forma y estilo a los elementos HTML de la página. Aprovechando su extensa documentación, se ha seleccionado componentes predefinidos que, con pequeñas modificaciones, se han integrado de manera sencilla e intuitiva en el código.

3.1.3 Jquery

jQuery es una biblioteca de JavaScript que simplifica la manipulación, animación y realización de peticiones asíncronas en el desarrollo de aplicaciones web. Esta característica es especialmente valiosa para crear una interfaz de usuario fluida en la implementación de un chat en tiempo real. Mediante la tecnología AJAX, podemos realizar solicitudes a diferentes rutas de manera asíncrona para mostrar los mensajes en pantalla en respuesta a eventos. [10]

En este contexto, jQuery ha sido utilizado principalmente para escuchar eventos relacionados con el envío de mensajes, permitiéndonos realizar solicitudes en segundo plano sin abandonar la vista actual. Esta funcionalidad es esencial en la implementación de un chat, como el que se ha desarrollado en esta aplicación.

3.1.4 Json

JSON es un formato de intercambio de datos caracterizado por su estructura de texto plano y legibilidad sencilla. Es ampliamente utilizado en aplicaciones web para transmitir información y es común en la comunicación entre distintas aplicaciones y servicios en Internet. Aunque se origina en el lenguaje de programación JavaScript, es independiente y es compatible con diversos lenguajes de programación. [11]

En esta aplicación, se ha empleado JSON para intercambiar datos entre la API de WhatsApp y la aplicación. Las solicitudes realizadas incluyen cuerpos de datos en formato JSON para gestionar la información enviada y recibida. Además, se ha utilizado JSON para el intercambio de datos internamente en la aplicación, permitiendo la transferencia de información entre diversas vistas y controladores.

En el proceso de programar la recepción de Webhooks, ha sido fundamental manejar adecuadamente los objetos JSON, ya que contienen información sobre los mensajes recibidos y su estado. Además, se realizan respuestas correspondientes para situaciones como la configuración de los Webhooks.

Para la gestión de elementos JSON, se ha hecho uso de una página web que consiste en un visor de este tipo de objetos, en este caso la aplicación se llama jsonviewer y como se muestra en la Figura 7 facilita la lectura de este tipo de objeto ya que a veces puede ser un tanto complejo su comprensión en formato de texto.

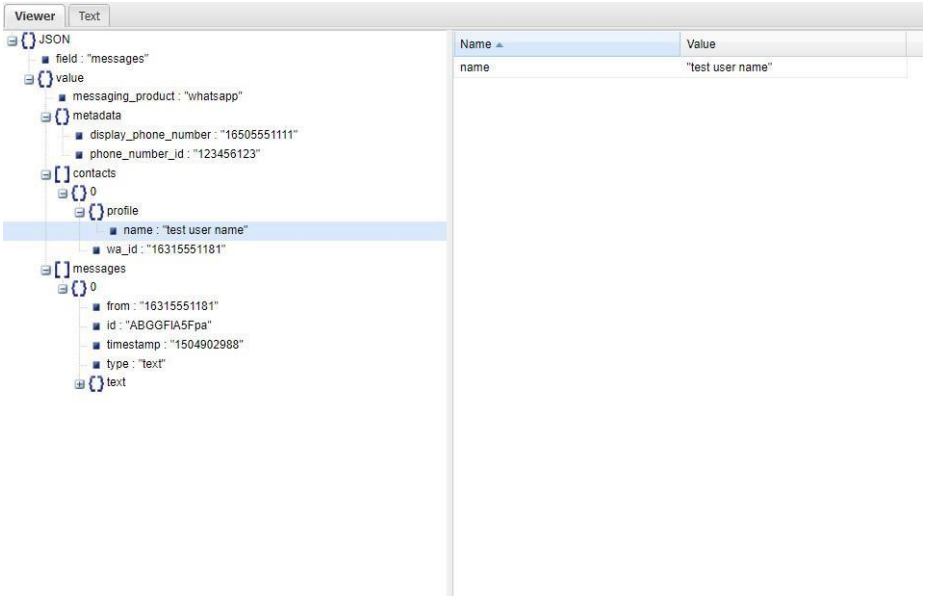


Figura 7 Elemento JSON tipo mensaje recibido por Webhook visto desde el visor jsonviewer

3.1.5 Pusher

Pusher es un servicio de mensajería en tiempo real basado en la nube que facilita la comunicación bidireccional instantánea entre clientes y servidores a través de WebSocket. Ofrece una infraestructura escalable y confiable para transmitir datos en tiempo real mediante diversos canales y eventos, gracias a su arquitectura de publicar/suscribir. [12]

En este proyecto, Pusher ha sido una pieza fundamental para recibir mensajes en tiempo real. En el lado del servidor, se crea una instancia proporcionando los datos necesarios y se envía un evento al canal correspondiente. Por otro lado, en el lado del cliente, se inicializa un objeto que se suscribe a un canal, permitiendo la escucha de eventos para así poder recargar la página y mostrar el mensaje recibido.

Pusher ofrece una plataforma, ilustrada en la figura 8, desde la cual se puede observar el canal creado, las conexiones realizadas en el día y los mensajes enviados. Dado que este canal es utilizado para las conexiones en tiempo real, permite rastrear los mensajes enviados y recibidos por la aplicación.

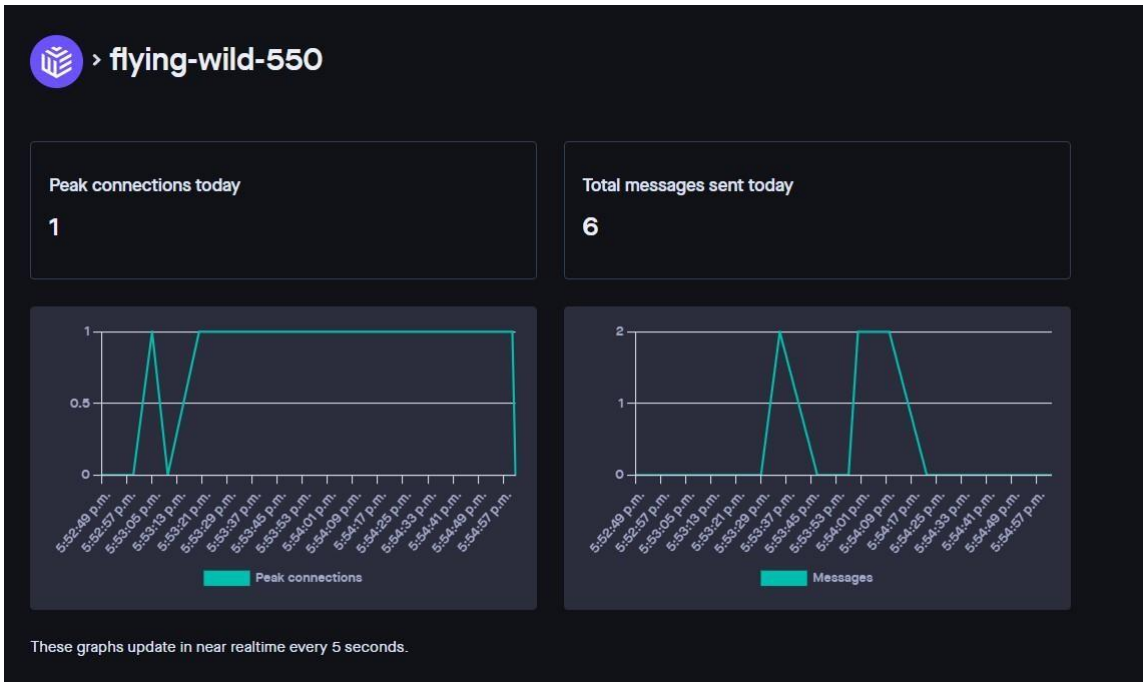


Figura 8 Panel de control plataforma Pusher

3.2 Herramientas

3.2.1 Visual Studio Code

Visual Studio Code es un editor de código fuente extremadamente popular y ampliamente utilizado en proyectos de desarrollo. Su combinación de ligereza y alta capacidad de personalización lo convierte en la opción preferida para numerosos programadores. [13]

Una de las ventajas más destacadas de Visual Studio Code es su amplia compatibilidad con una gran variedad de lenguajes de programación. Esto permite trabajar en diversos proyectos sin necesidad de cambiar de herramienta. Además, cuenta con un ecosistema de extensiones sólido que expande su funcionalidad y lo adapta a las necesidades específicas de cada proyecto.

En el contexto del proyecto en cuestión, Visual Studio Code se ha utilizado como el editor de código principal. Su versatilidad en términos de lenguajes de programación, así como la disponibilidad de extensiones especializadas, han facilitado la generación y edición eficiente de código. Además, la presencia de un terminal integrado ha agilizado la ejecución de comandos de Laravel, agilizando la creación de nuevos componentes y mejorando la productividad del desarrollo.

3.2.2 Postman

Postman es una herramienta crucial para el desarrollo de APIs, permitiendo una prueba y desarrollo rápidos y sencillos de estas interfaces. Con Postman, es posible enviar solicitudes HTTP y visualizar las respuestas de manera intuitiva, sin necesidad de escribir código. [14]

En este proyecto, Postman ha sido una herramienta esencial, especialmente gracias a las colecciones públicas disponibles en la plataforma. En particular, encontramos una colección dedicada a las peticiones manejadas por la API de WhatsApp Business. Aprovechando esta colección y realizando ajustes mínimos en las configuraciones por defecto, se ha podido evaluar el funcionamiento de la aplicación.

Además, la colección proporcionaba una documentación concisa y comprensible, abordando detalles como EndPoints, parámetros, encabezados y manejo de respuestas esperadas. Cuando se realizaban peticiones incorrectas, la aplicación brindaba informes de error detallados y señalaba los puntos problemáticos, lo que permitía una respuesta rápida y eficiente para abordar cualquier problema.

3.2.3 Xampp

XAMPP es un programa de software de código abierto y gratuito que ofrece un entorno de servidor local para la creación y desarrollo de aplicaciones, especialmente para aplicaciones web. Además, es una herramienta fundamental para llevar a cabo pruebas en entornos locales, gracias a las numerosas aplicaciones que incluye. [15]

En este proyecto, se ha aprovechado la herramienta MySQL de XAMPP, que es la encargada de iniciar el servidor de bases de datos, asignar recursos y crear conexiones necesarias. Para trabajar con una base de datos similar a la que se encuentra en el entorno en la nube, primero se despliega la estructura de la base de datos en el equipo de desarrollo local. Esto nos permite probar el correcto funcionamiento antes de implementar en la nube.

Además, se ha utilizado la herramienta Apache, que proporciona un servidor web local. Esto permite probar y depurar el código antes de subirlo al entorno en la nube. Para usar el código generado en la aplicación con XAMPP, es necesario agregar la carpeta raíz del proyecto en una ubicación específica dependiendo del proceso de instalación del programa. Luego, al acceder a "http://localhost:80/whatsapp/server.php" desde un navegador, se visualizará la aplicación web que en desarrollo.

3.2.4 HeidiSQL

HeidiSQL es un software gratuito con el propósito de ser fácil de aprender. HeidiSQL permite visualizar y editar datos y estructuras en equipos que utilizan uno de los sistemas de bases de datos como MariaDB, MySQL y PostgreSQL entre otros. Inventado en 2002 por Ansgar, HeidiSQL es una de las herramientas más populares a nivel mundial para MariaDB y MySQL. [16]

En este proyecto, se ha utilizado HeidiSQL como visor de la base de datos SQL para agilizar la verificación de eventos relacionados con la interacción en la base de datos, como la inserción, modificación o eliminación de registros. Además, al comienzo del proyecto, esta herramienta nos facilitó la creación de la base de datos a través de su interfaz gráfica, lo que nos permitió agregar de manera simple y eficiente las credenciales de seguridad y la información de conexión.

3.2.5 Composer

Composer es una herramienta de gestión de paquetes para PHP que facilita la administración de dependencias y componentes, incluyendo aquellos propios del framework Laravel. Además de mantenerlos actualizados, Composer permite la inclusión de bibliotecas de terceros, almacenando toda esta información en el archivo "composer.json". [17]

Composer ha sido esencial tanto para instalaciones locales como en la plataforma en la nube. Al crear un nuevo proyecto de Laravel, la documentación recomienda el uso de esta herramienta. Primero, ejecutamos el comando *composer global require laravel/installer* y luego *composer create-project --prefer-dist laravel/laravel:^7.0 whatsapp* en la carpeta donde se inicia el proyecto.

Además, se han instalado a través de Composer varios paquetes externos que han facilitado la integración de la API con la aplicación. Un ejemplo es el paquete "pusher/pusher-php-server", que fue mencionado anteriormente. Otro paquete es "laravel/ui", que proporciona funcionalidades de interfaz de usuario basadas en Bootstrap, que ha sido usado con el propósito de generar una pantalla de registro agradable de manera ágil y rápida.

3.2.6 GitHub

GitHub es una plataforma en la nube que utiliza el sistema de control de versiones Git y se emplea para alojar proyectos. Su función principal es rastrear el código generado, lo que permite agregar nuevas funcionalidades y corregir errores mediante la creación de diferentes versiones. [18]

Con el fin de facilitar la colaboración entre equipos dispersos, se ha creado un repositorio en esta plataforma para cargar el código. Además, este código se ha descargado en otros equipos utilizando Git, asegurando una sincronización adecuada. Adicionalmente, GitHub proporciona la opción de cargar directamente el código en la nube a través de su plataforma.

Mediante esta herramienta, se ha mantenido un seguimiento detallado del progreso de la aplicación. Esto se ha logrado a través de su interfaz gráfica y de los mensajes asociados a los "commits". Estos mensajes ofrecen información precisa sobre los cambios realizados en el código, contribuyendo a mantener un registro claro y organizado del desarrollo del proyecto.

3.2.7 Plesk

Plesk es una plataforma de administración de servidores web que permite gestionar y controlar sitios web alojados en la nube. Ofrece una interfaz gráfica intuitiva y fácil de usar para la administración de aplicaciones.

Para llevar la aplicación a internet y asegurar su funcionamiento en cualquier navegador, se ha usado un servidor basado en el sistema operativo Linux alojado en la nube. Con esta herramienta, se ha subido el código de la aplicación, que incluye integración con Git para facilitar el despliegue del código. Además, Plesk cuenta con un administrador de bases de datos que simplifica la gestión de la base de datos del servidor en la nube.

La plataforma también brinda una conexión SSH al servidor, lo que ha sido esencial en este proyecto. Al utilizar Laravel, en ocasiones es necesario generar migraciones para crear bases de datos o ejecutar comandos que interactúen con la aplicación para implementar diversas funcionalidades. La capacidad de interactuar con el servidor a través de SSH ha sido fundamental para realizar estas tareas de manera eficaz. [19]

Capítulo 4. Aplicación

4.1 Análisis

Tras la investigación y la comprensión de la API con la que se va a integrar la aplicación, se han obtenido las diferentes funcionalidades que se ofrecen, una vez vista se ha hecho una selección de las que han sido consideradas las más adecuadas para la implementación, cuando se habla de la API en la nube de WhatsApp Bussines se hace referencia principalmente al envío y la recepción de mensajes.

En cuanto a los mensajes enviados, existen dos tipos: los convencionales, que representan la forma principal de comunicación en WhatsApp, y los de tipo plantilla, que son necesarios para iniciar conversaciones y agilizar el envío de mensajes predefinidos, incluyendo preguntas frecuentes o situaciones repetitivas. Para los mensajes convencionales, se incluyen:

- Mensajes de texto
- Mensajes de reacción
- Mensajes de contenido Multimedia
- Mensajes de ubicación
- Mensajes de contactos
- Mensajes interactivos

En cuanto a los mensajes de tipo plantilla, estos pueden ser:

- Plantillas basadas en mensajes de texto
- Plantillas basadas en contenido multimedia
- Plantillas de mensajes interactivas

Para la recepción de mensajes, es fundamental considerar el uso de Webhooks. Al configurar un Webhook desde la plataforma de WhatsApp Business, es importante tener en cuenta que este recibirá mensajes de diferentes tipos, que se dividen en tres categorías:

1. Mensajes convencionales, que abarcan los mismos tipos de mensajes que se pueden enviar.
2. Mensajes de estado, que indican el estado de los mensajes enviados y recibidos, permitiendo un seguimiento detallado de la conversación, tanto para el usuario como para fines de registro.
3. Mensajes de error, que se generan si surgen problemas en la comunicación entre la API y la aplicación, afectando su funcionamiento.

[20]

Las funcionalidades clave que se implementarán en la aplicación son: envío de mensajes de texto y contenido multimedia, así como el uso de distintos tipos de plantillas. También se considerará la recepción de mensajes en todas sus variantes.

El código de la aplicación se desarrollará de manera modular, facilitando la integración de nuevas funcionalidades en el futuro. Se buscará generar bloques de código comprensibles que simplifiquen las implementaciones posteriores.

4.2 Base de datos

Tras analizar las funcionalidades de la aplicación, se procedió al diseño de la base de datos cuya estructura se muestra en la Figura 9. En esta figura se visualizan diversos campos que se utilizarán para gestionar los datos de la aplicación.

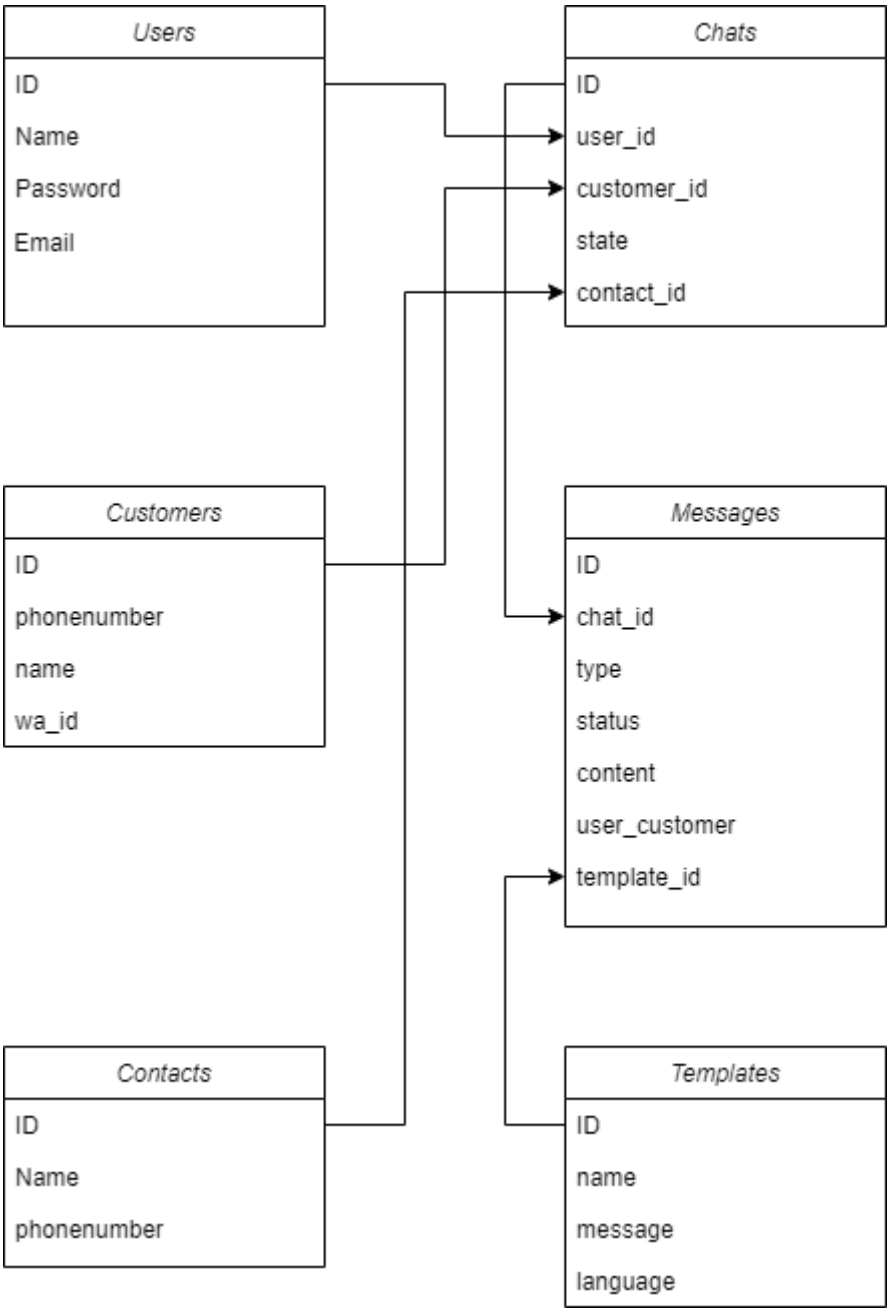


Figura 9 Diagrama base de datos de la aplicación

Los campos que se presentan en la figura son los siguientes:

- Users
  - ID: Identificador del usuario.
  - name: Nombre del usuario en formato de cadena.
  - password: Hash de la contraseña del usuario almacenado como cadena.
  - email: Matriz que guarda los correos electrónicos asociados al usuario.

- Chats
  - ID: Identificador del chat.
  - user_id: Identificador que hace referencia a la tabla Users.
  - customer_id: Identificador que hace referencia a la tabla Customers.
  - contact_id: Identificador que hace referencia a la tabla Contacts.
  - state: Cadena para almacenar el estado del chat.

- Messages
  - ID: Identificador del mensaje.
  - chat_id: Identificador que hace referencia a la tabla Chats.
  - type: Cadena que almacena el tipo de mensaje.
  - status: Cadena que guarda el estado del mensaje.
  - content: Contenido del mensaje almacenado como cadena.
  - user_customer: Valor booleano para identificar al remitente del mensaje.
  - template_id: Identificador que hace referencia a la tabla Templates.

- Customers
  - ID: Identificador del cliente.
  - name: Nombre del cliente como cadena.
  - phonenumber: Número de teléfono del cliente como cadena.
  - wa_id: Identificador de la cuenta de WhatsApp Business en formato de cadena.

- Contacts
  - ID: Identificador del contacto guardado.
  - name: Nombre del contacto almacenado como cadena.
  - phonenumber: Número de teléfono del contacto como cadena.

- Templates
  - ID: Identificador de la plantilla.
  - name: Nombre de la plantilla de meta en forma de cadena.
  - message: Contenido de la plantilla almacenado como cadena.
  - lenguaje: Idioma en que esta disponible la plantilla.

Una vez completado el diseño de la base de datos, se procedió a generar los modelos en el código para representar estas relaciones.

4.3 Envío de mensajes

Para enviar mensajes desde la aplicación se debe considerar los tiempos de conversaciones mencionados previamente y el uso de platillas para iniciar o reiniciar una conversación, además de los tipos de mensajes que existen.

En primer lugar, es necesario crear algunas plantillas desde la plataforma de Meta para iniciar una conversación desde la aplicación. Luego, se deben incorporar las funcionalidades de envío de mensajes correspondientes a los tipos seleccionados en el código, y finalmente, se va a mostrar en tiempo real estos mensajes en pantalla para seguir el flujo de la conversación.

4.3.1 Creación de plantillas

Para crear una plantilla desde el panel de control se la plataforma de WhatsApp Bussines como se muestra en la Figura 5, hay un botón donde indica administrar plantillas de mensajes, desde aquí se redirigirá al panel de administración de plantillas, como se muestra en la Figura 10, desde aquí se inicia el proceso de creación de una plantilla.

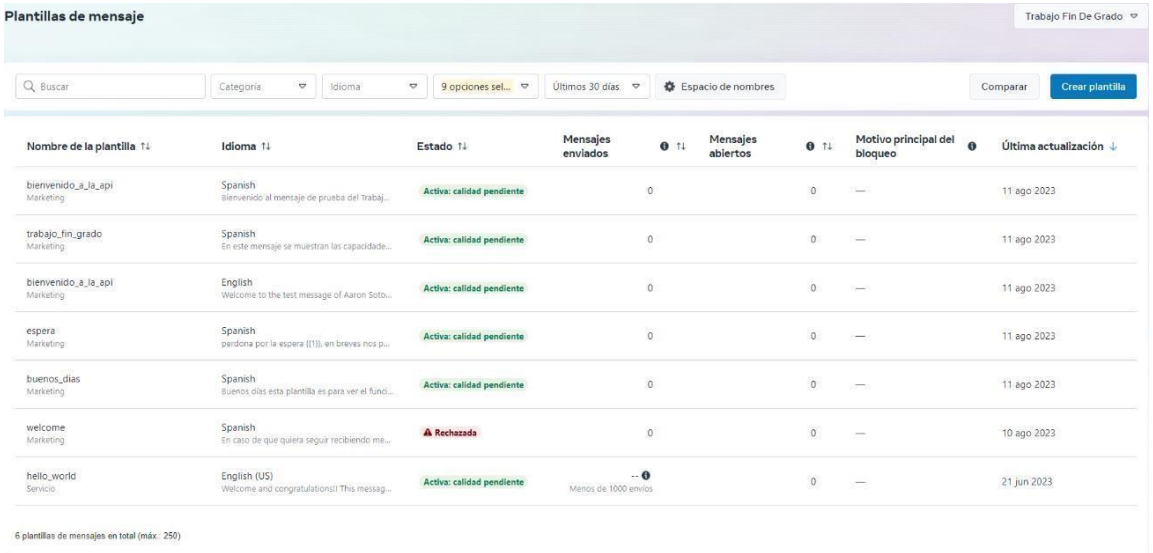


Figura 10 Panel de administración de plantillas

Para continuar con la creación de las plantillas, es necesario seleccionar “Crear plantilla”. Se abrirá una ventana inicial en la que se deberá especificar la categoría de la plantilla, su nombre e idioma. Es crucial seleccionar la categoría adecuada para superar con éxito el proceso de revisión establecido por la empresa Meta. En este caso, la categoría 'Marketing' es la apropiada.

Una vez que se hayan completado los campos según se muestra en la Figura 11, se accede al editor de plantillas. En este apartado, es recomendable completar cada campo del mensaje que se pretende enviar de manera adecuada. El único campo obligatorio es el cuerpo del mensaje, pero es beneficioso añadir botones para que el destinatario pueda responder de manera sencilla. Conforme se rellenan los campos, como se muestra en la Figura 12, se muestra una vista previa de cómo quedará la plantilla, permitiendo ver cómo se presentará al cliente.

Ya creada la plantilla y declarada como “Activa” por Meta, ya se puede utilizar el mensaje recién creado en la aplicación. Para enviar este tipo de mensaje a través de la API, solo se requieren dos campos: el nombre de la plantilla y el idioma en el que se creó. Esto se refleja en la tabla “template”, tal como se muestra en la Figura 9, donde existen los campos “name” y “lenguaje”, que hacen referencia a estos dos valores necesarios para su uso.

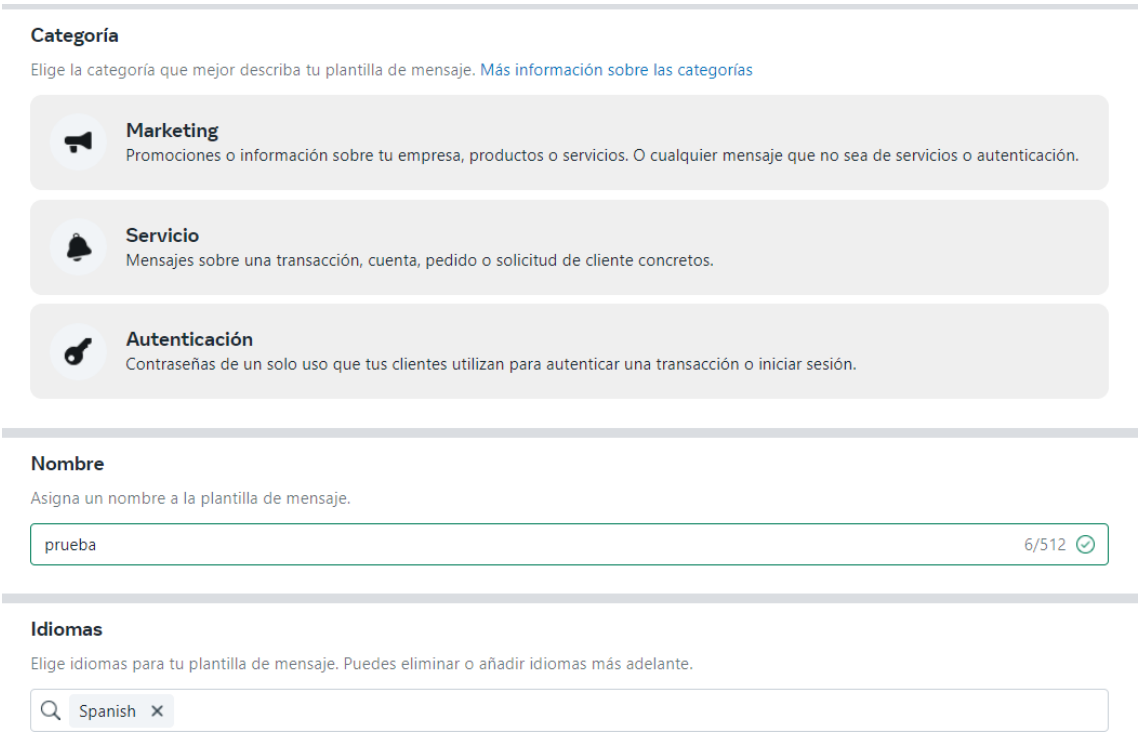


Figura 11 Configuración previa de una plantilla de WhatsApp

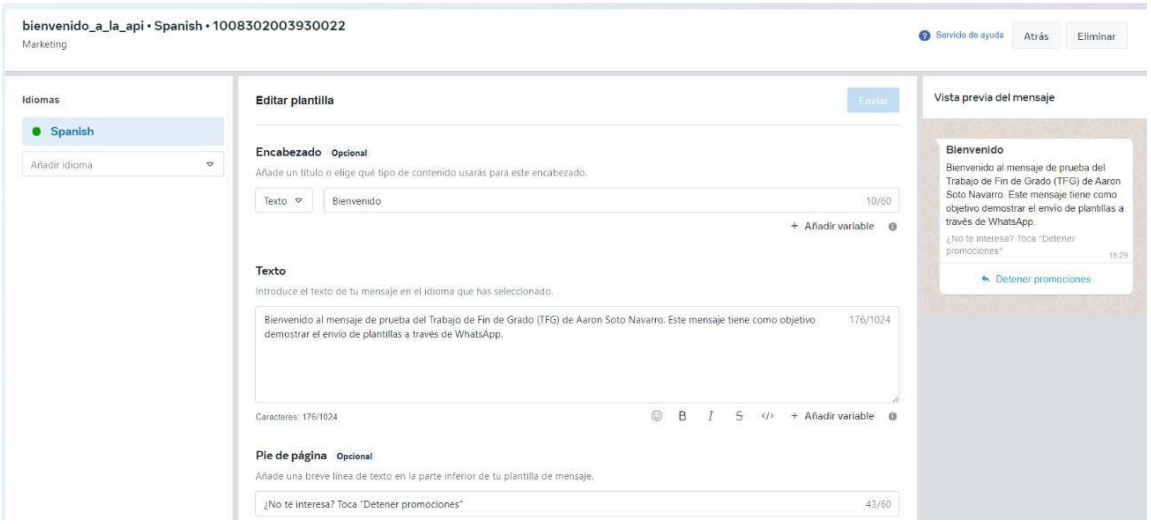


Figura 12 Configuración del cuerpo de una plantilla de WhatsApp

4.3.2 Implementación del código

El proceso para implementar el envío de mensajes mediante código en la aplicación incluye varios pasos clave que garantizan su correcto funcionamiento y una interacción fluida con la base de datos y la API de WhatsApp. Para llevar a cabo esto, se utiliza Artisan, el CLI proporcionado por Laravel para generar los componentes necesarios en esta implementación.

En primer lugar, se crea un componente de migración ejecutando el comando *php artisan make:migration create_messages_table*. Esta migración describe la estructura de la tabla "messages" en la base de datos, definiendo los campos necesarios para almacenar los mensajes, como el contenido, el remitente, la marca de tiempo, entre otros.

A continuación, se ejecuta la migración mediante el comando *php artisan migrate*, lo que crea la tabla "messages" en la base de datos de acuerdo con las especificaciones definidas en la migración.

Seguidamente, se define el modelo "Message" que representa la tabla "messages" en la base de datos. Al utilizar el comando *php artisan make:model Message*, se crea el modelo y se establecen las relaciones necesarias con otras tablas de la base de datos, lo que facilitará la gestión e interacción con los mensajes en la aplicación.

Posteriormente, definimos el modelo "Message" a través del comando *php artisan make:model Message*. Este modelo facilita el manejo y relación de los mensajes con otras partes de nuestra base de datos.

Con la base de datos lista, avanzamos hacia la interacción con la API de WhatsApp. Para ello, creamos el controlador "WhatsappMessageController" mediante el comando *php artisan make:controller WhatsappMessageController*. Este controlador es la pieza clave, ya que alberga la lógica necesaria para la comunicación con la API. Gracias a su diseño modular, podemos adaptarnos a diferentes tipos de mensajes y, si surge la necesidad, integrar nuevos tipos con facilidad.

En el lado del cliente, las interacciones se basan en diversas vistas que permiten enviar mensajes. La selección de la vista adecuada depende del tipo de mensaje que se desee transmitir, y todas están organizadas de manera accesible en un directorio específico.

Una vez generada la estructura con la que se va a proceder al envío de mensajes se han implementado una serie de funciones para el procesado de los mensajes a enviar, dependiendo del tipo de mensajes se hará llamada a una función u a otra.

Dentro de la funcionalidad de envío de mensajes por parte del cliente, es posible distinguir dos modalidades principales:

1. Envío desde el chat en tiempo real: En esta modalidad, los usuarios interactúan en una interfaz de chat en vivo. Gracias al uso de jQuery, los mensajes se envían y reciben sin necesidad de recargar la página o cambiar de vista, ofreciendo una experiencia más dinámica y fluida.
2. Envío desde vistas específicas: Estas vistas están diseñadas para el envío de mensajes más concretos y no muestran el intercambio en tiempo real. Aquí, el proceso de envío se realiza a través de formularios HTML. Una vez que el usuario envía el mensaje, este se dirige hacia rutas específicas que han sido previamente establecidas en el archivo "web.php", encargado de gestionar dichas peticiones.

Estas peticiones se encargan de recopilar información que necesitara el lado del servidor para procesar la información y mandar dichos mensajes.

Cuando el servidor recibe una petición, esta viene en forma de un objeto JSON que contiene la información del mensaje. A través de consultas a la base de datos, el servidor obtiene la información adicional requerida para el envío. Con estos datos, se configura un nuevo objeto JSON específicamente estructurado para la API de envío de mensajes. A continuación, se ejecuta

una petición POST hacia la API, la cual responde con un estado sobre el resultado del envío. Si el mensaje se envía correctamente, se registra en la base de datos con todos los campos relevantes. Finalmente, el proceso retorna al flujo de operación original. Es relevante mencionar que esta metodología se aplica tanto a mensajes basados en plantillas como a mensajes estándar

El tratamiento de los distintos tipos de mensajes varía ligeramente en su inicio y finalización. Para los mensajes de texto estándar, se utiliza una función AJAX que remite el mensaje al controlador "WhatsappMediaController". Esta función específica, denominada "sendText", Figura, se encarga de enviar mensajes de tipo texto. Al recibir el mensaje, "sendText" consulta la base de datos para determinar a qué conversación corresponde el mensaje y a qué cliente está destinado. Una vez que se han recopilado estos datos, se invoca a una función genérica llamada "sendPostRequest".

A la función "sendPostRequest" se le pasa el cuerpo del mensaje que se desea enviar con todos sus detalles. Esta función devuelve una respuesta basada en lo que la API ha retornado. Si todo ha ido bien, devuelve un código 200; en caso de error, se proporciona un código de error diferente. Según la respuesta obtenida, se invocará a otra función encargada de registrar el mensaje en la base de datos llamada "saveMessage". Tras este proceso, se remite un objeto JSON a la función AJAX original que actuará en función de la respuesta recibida.

En lo que respecta a la función "sendPostRequest", procesa el objeto pasado como argumento y genera el objeto adecuado para la solicitud POST. Los parámetros de la cabecera, como el token de autenticación, se extraen del archivo ".env". La URL utilizada es una variable global que abarca todo el controlador. Una vez configurado, se procede al envío del mensaje.

El proceso de envío de plantillas comparte similitudes con el de los mensajes de texto, aunque inicia de manera distinta. En primer lugar, es necesario añadir la plantilla deseada dentro de la aplicación. Una vez configurada, se habilita la opción de enviarla a un cliente a través de un formulario.

Dado que este procedimiento no requiere operaciones en segundo plano, se emplean formularios HTTP estándar. En estos, se introducen los campos necesarios para la plantilla y, posteriormente, se invoca a la función "sendTemplate" del controlador "WhatsappMessageController". Aunque esta función opera de forma parecida a la de los mensajes de texto convencionales, su respuesta difiere: en lugar de retornar un objeto JSON, redirige al usuario a la vista de gestión de plantillas.

El proceso de envío de mensajes multimedia es algo más complejo y puede realizarse de dos maneras: mediante el envío del contenido por URL o utilizando un identificador. En el caso del envío por URL, es tan sencillo como enviar un mensaje de texto. Sin embargo, para el envío utilizando un identificador, se necesita primero obtener dicho identificador mediante una petición POST a la API.

Para gestionar estos mensajes multimedia, se ha creado un controlador separado llamado "WhatsappMediaController". En este proceso, se envía un mensaje tipo POST a la API, subiendo el archivo correspondiente. Cabe destacar que el archivo debe cumplir ciertas especificaciones: debe tener un formato MIME específico y no puede superar un tamaño máximo determinado. Por ello, se realiza una petición específica al EndPoint "/media", adjuntando el archivo que se desea enviar. Estos archivos pueden ser de varios tipos: audio, imagen, video, sticker o documento.

Desde una vista destinada a la gestión de mensajes multimedia, se invoca la función "store" del controlador. Esta función verifica primero el tipo de archivo y confirma si cumple con las especificaciones requeridas. Una vez validado el archivo, se realiza la petición POST y se obtiene el identificador correspondiente. Con este identificador en mano, se puede llamar a las funciones específicas de "WhatsappMessageController" dedicadas al envío del tipo de archivo en cuestión.

Al no mostrarse en tiempo real en este desarrollo, tras el envío, la función retorna y redirige al usuario a una vista, similar a cómo se gestiona con las plantillas.

```
0 references | 0 overrides
public function sendText(Request $request, $chat_id)
{
    Log::info("Intentando enviar un mensaje de texto via WhatsApp");

    $chat = Chat::findOrFail($chat_id);
    $phoneNumber = $chat->customer->phone_number;
    $bodyText = $request->input('message_content');

    $data = [
        "messaging_product" => "whatsapp",
        "recipient_type" => "individual",
        "to" => $phoneNumber,
        "type" => "text",
        "text" => [
            "preview_url" => false,
            "body" => $bodyText
        ]
    ];

    $response = $this->sendPostRequest($data);

    if ($response['status'] == 200) {
        $this->saveMessage($chat_id, $bodyText, 'text');
        return response()->json(['status' => 'success'], 200);
    } else {
        return response()->json(['error' => 'Error al enviar el mensaje'], 500);
    }
}
```

Figura 13 Función "SendText" controlador WhatsappMessageController.php

4.4 Recepción de mensajes

La recepción de mensajes se lleva a cabo a través de los Webhooks, lo que implica configurarlos tanto en la plataforma de WhatsApp como en la aplicación. Desde el punto de vista de la plataforma de WhatsApp, es esencial especificar qué tipos de objetos se desean recibir como parte de la comunicación.

Desde el lado del código de la aplicación, se deben implementar procesos para gestionar y procesar los campos de los objetos que se pretenden recibir de los Webhooks. Esto asegura que la aplicación esté preparada para manejar eficazmente los datos entrantes y responder de manera adecuada.

Además de los mensajes estándar, es importante tener en cuenta los mensajes de estado de la aplicación. En ocasiones, puede ocurrir que un mensaje no se pueda enviar debido a políticas o restricciones de la API. En tales casos, la API enviará un mensaje de error que indicará que el mensaje no pudo ser entregado correctamente. Por lo tanto, en esta parte del proceso, también se deben abordar los escenarios en los que el envío de un mensaje ha resultado erróneo.

4.4.1 Configuración Webhook

La configuración de un Webhook en la plataforma de WhatsApp Business se lleva a cabo mediante la creación de un EndPoint en la aplicación, capaz de procesar peticiones POST y GET. Este proceso es esencial para habilitar la recepción y manejo de mensajes y eventos desde WhatsApp hacia nuestra aplicación.

Para configurar el Webhook desde la plataforma de WhatsApp Business, se debe acceder a la sección designada para los Webhooks. Una vez allí, el primer paso es elegir el tipo de elemento al que se desea suscribir. En este caso, la opción adecuada es “WhatsApp Business Account”, ya que desde allí podremos seleccionar las opciones necesarias para recibir tanto mensajes como eventos de diversos tipos, que proporcionan información valiosa sobre el estado de la cuenta de WhatsApp, entre otras funcionalidades.

Al acceder a la ventana de configuración, nos encontramos con un botón destinado a la suscripción del objeto. Al hacer clic en dicho botón, se abrirá una nueva ventana solicitando que indiquemos el EndPoint de nuestra aplicación, junto con un desafío para comprobar la corrección del EndPoint que estamos añadiendo. Es importante mencionar que este identificador se encuentra en el archivo “.env”, donde se guardan las variables de entorno para el uso general de la aplicación.

Una vez completada la información requerida, al pulsar el botón de verificar, se enviará una petición GET, obteniendo ciertos campos que permitirán validar si nuestra aplicación ha resuelto el desafío de manera exitosa. De esta forma, se confirmará la suscripción al Webhook, y estaremos listos para recibir mensajes y eventos desde WhatsApp.

Dentro de la ventana de configuración, tenemos la posibilidad de especificar los campos que deseamos recibir, así como seleccionar la versión de la API que se utilizará para las interacciones. Además, se nos proporciona un ejemplo del tipo de mensaje que recibiremos en el EndPoint de la aplicación, presentado en formato JSON. Será fundamental procesar adecuadamente estos mensajes mediante el uso de código, permitiéndonos realizar diversas acciones según nuestras necesidades.

Un detalle relevante durante el proceso de configuración es asegurarnos de seleccionar el campo “messages” como prioridad principal. De esta manera, estaremos configurando el Webhook para recibir tanto mensajes como estados asociados a dichos mensajes. Asimismo, para verificar el estado y calidad de las plantillas utilizadas, contamos con los campos “message_template_quality_update”, “message_template_status_update” . Estas opciones nos permitirán verificar el estado de las plantillas de una forma más rápida, por lo que habría que dejarlo como en la Figura 14.

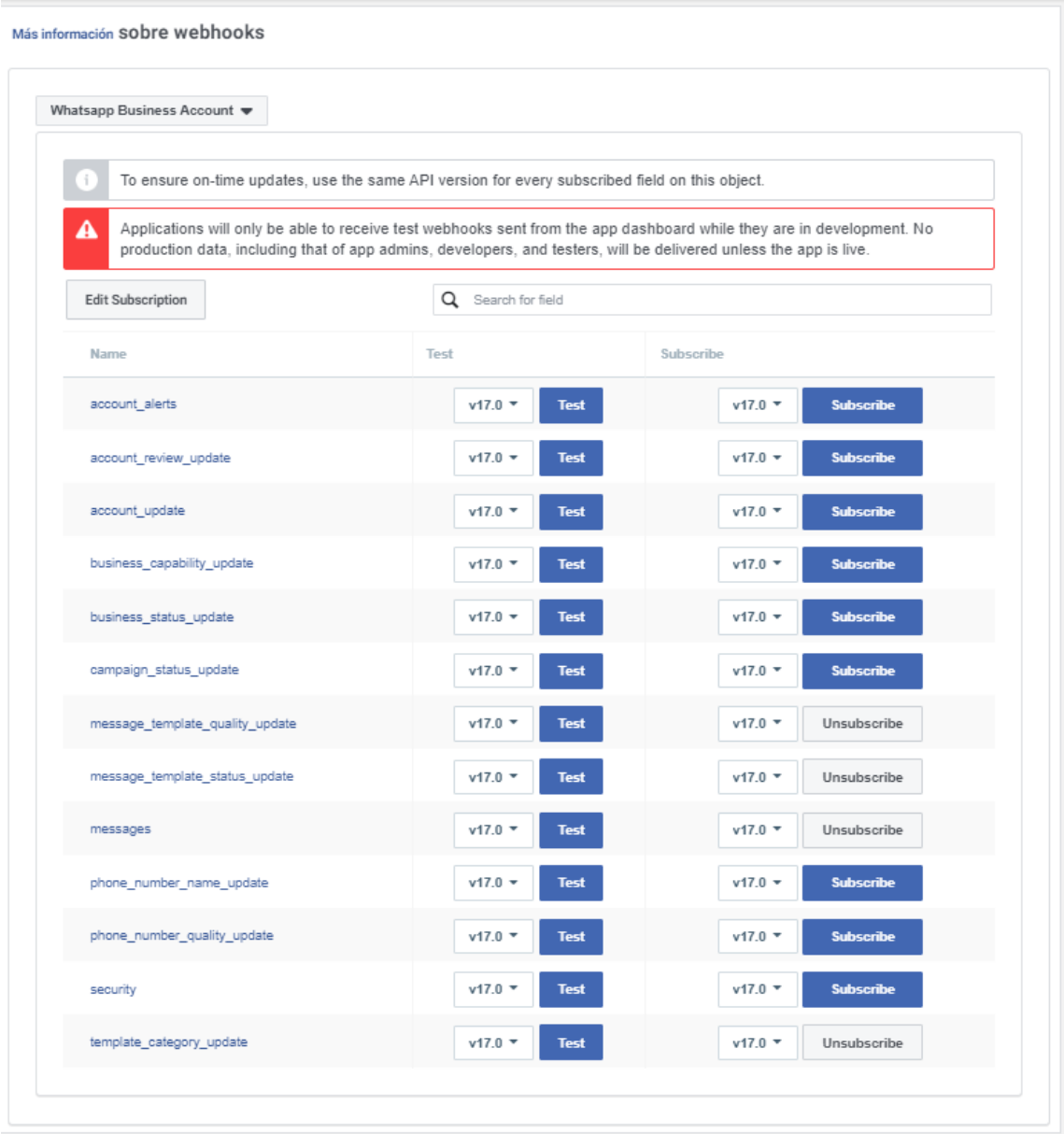


Figura 14 Eventos de Webhook a recibir en la aplicación

4.4.2 Implementación del código

Para la implementación de esta parte del código también ha sido necesario el uso del CLI de Laravel, para la creación de un controlador que procese la información relacionada con el Webhook se ha usado el comando *php artisan make:controller WebhookController*, Dentro de este controlador se encuentra una función central que es responsable de gestionar las peticiones entrantes, ya sea mediante el método GET o POST. La configuración de cómo se manejarán estas peticiones está definida en el archivo de rutas "web.php".

Cuando una solicitud llega al controlador y es de tipo GET, se lleva a cabo una verificación del mensaje enviado desde la plataforma de WhatsApp. Esta verificación es esencial para garantizar que el Webhook sea válido y seguro. Se realiza una comprobación inicial de los campos "hub_mode", "hub_verify_token" y "hub_challenge". Si estos campos están presentes y coinciden con los valores esperados, incluido el token de verificación almacenado en una variable

de entorno de la aplicación, se devuelve una respuesta HTTP con un código 200. Este código indica que la suscripción al Webhook se ha realizado con éxito. Sin embargo, si alguno de estos parámetros no coincide con lo esperado, se devuelve un código de error 403. Este código indica que la petición no ha sido realizada adecuadamente, lo que ayuda a prevenir posibles fraudes y asegura que solo las fuentes legítimas puedan suscribirse a la aplicación.

En el caso de las solicitudes POST, que contienen los datos reales de los mensajes, se lleva a cabo un proceso más detallado. Primero, el JSON recibido se convierte en un array para poder manejar y analizar los diferentes tipos de solicitudes que llegarán a esta función. Luego, se procede a identificar el tipo de mensaje que se ha recibido.

En función del tipo de mensaje, el código verifica su tipo y actúa en consecuencia. Cuando se trata de un mensaje en el que el tipo se recibe en primer lugar de manera general, consulta la base de datos para determinar si existe el chat del que se ha recibido el mensaje. Una vez realizada la comprobación, si el chat no existe, se crea, y si existe, se verifica a qué cliente pertenece para añadirlo posteriormente en la base de datos. Independientemente del tipo de mensaje, se almacena en la base de datos, ya sea como un mensaje nuevo o de un chat existente.

Otro tipo de mensajes que llegarán son los mensajes de estado. En este tipo de mensaje se proporcionará información diversa sobre el funcionamiento de la aplicación. Dentro de este tipo de objeto se encuentran tres variantes de mensajes.

La primera información se refiere a los mensajes enviados. Cuando se envía un mensaje, llega a este punto final con su estado, que puede ser "enviado", "recibido" o "leído", al igual que en la aplicación móvil. Estos tres estados nos indican la interacción con la persona a la que se le ha enviado el mensaje, permitiéndonos llevar un seguimiento de la conversación.

Luego, también se encuentra información dentro de estos mensajes acerca de quién ha iniciado la conversación. En el caso de que sea el primer mensaje, o si la conversación ya ha sido iniciada anteriormente, se controlan las limitaciones de la API.

Adicionalmente, se efectuará el procesamiento de mensajes de error vinculados a la API en este controlador, y los posibles errores serán registrados en un archivo de LOGS, permitiendo un análisis detallado de las interacciones con la API.

4.5 Estados

Para mostrar de forma adecuada los mensajes y conversaciones que se han ido generando según se ha encontrado en funcionamiento la aplicación, se han usado estados, en ellos se muestra en cada parte de la ventana principal de la aplicación de forma adecuada, estos estados son los que permiten saber como se encuentran los distintos mensajes y conversaciones.

4.5.1 Estados de los mensajes

Para la gestión de los mensajes, como se muestra en la Figura 9 en la tabla “messages”, existe el campo “status”. Este campo desempeña un papel importante en el seguimiento y tratamiento de los mensajes, tanto los enviados como los recibidos. Dependiendo de la situación, este campo determinará la apariencia del mensaje, en este caso el estado no variara dependiendo de si es plantilla o no.

Cuando se envía un mensaje se le asigna el valor “send”, indicando que el mensaje ha sido enviado correctamente mediante una petición POST a la API de WhatsApp, una vez el mensaje ha sido enviado satisfactoriamente, es posible que llegue un mensaje de respuesta por parte de la API, este mensaje ira a donde se ha configurado el Webhook, y será procesado dependiendo del cuerpo que tenga el mensaje actualizando el campo “status” dependiendo de lo que indique, los casos que se pueden dar a continuación son que el mensaje ha llegado al dispositivo de la persona a la que se ha enviado, por lo que se actualizara el campo a “send”, y el siguiente caso es que el usuario final haya leído el mensaje, por lo que se actualizara a “read”, ambos casos requieren de una interacción con el cliente final por lo que cabe la situación de que estos objetos no lleguen.

Cuando un mensaje se envía, se le asigna el valor “send”, indicando que ha sido correctamente enviado a través de una solicitud POST a la API de WhatsApp. Después del envío exitoso, es posible recibir una respuesta de la API, que se dirigirá al lugar configurado en el Webhook y se procesará según su contenido. Esta respuesta actualizará el campo “status” en función de su contenido. Los dos casos principales son los siguientes:

1. Si el mensaje ha llegado al dispositivo del destinatario, el campo “status” se actualizará a “send”.
2. Si el destinatario ha leído el mensaje, el campo “status” se actualizará a “read”.

Es importante destacar que estos dos casos requieren una interacción por parte del cliente final, por lo que es posible que estos estados no se alcancen en todos los escenarios.

Si en el proceso de realizar una solicitud POST surge algún tipo de error, se registra como un estado de “error”. El escenario más recurrente al que nos podemos enfrentar es el error 500, el cual suele apuntar a problemas de permisos y se comunica a través de un mensaje de respuesta. Esta clase de error puede originarse por un token caducado o por valores inapropiados en la solicitud misma. Con el propósito de agilizar la resolución de estas incidencias, hemos incorporado variables de entorno en todo el código. Esto permite que dichas variables sean utilizadas de manera ágil y sencilla en cualquier sección del programa para intercambiar valores sensibles de fallos.

Toda esta información sobre los estados de los mensajes se presenta de manera visual en la interfaz del chat. La representación gráfica del estado de cada mensaje se efectúa mediante íconos distintos, lo cual brinda una comprensión más sencilla del progreso de la conversación. Para lograr este efecto, se utiliza código PHP en la plantilla “chat.blade.php”, aprovechando bucles “foreach” para iterar sobre los mensajes y representar, mediante un booleano, de qué lado proviene el mensaje, permitiendo su ubicación en la parte adecuada de la conversación, cabe destacar que en el caso de que un mensaje se haya mandado de forma incorrecta este mensaje se perderá y habrá que mandarlo nuevamente.

4.5.2 Estados de las conversaciones

Por el funcionamiento de esta API, se ha implementado una gestión cuidadosa de las conversaciones para garantizar un uso apropiado de la aplicación y prevenir comportamientos inadecuados, ya que una vez pasada la venta de 24 horas desde que se ha recibido el último mensaje por parte del cliente ya no se podrá mandar mensaje a menos que sea una plantilla.

Para ellos al igual que en el apartado anterior si se revisa la Figura 9 se observa que en la tabla “chats” hay un campo llamado ‘sate’, este campo es el encargado de organizar las conversaciones distribuyéndolas adecuadamente en la vista principal de la aplicación.

Cuando se quiere iniciar una conversación con un cliente, debe comenzar con una plantilla, al enviar la plantilla, la conversación quedará registrada con el estado “send”, cuando esto ocurra en la conversación no será posible mandar ningún otro mensaje y habrá que esperar a recibir respuesta por parte del cliente.

Una vez la conversación ha sido respondida por parte del cliente, este campo será actualizado al estado “active”, lo que indica que ya ha sido contestado y por lo tanto se podrá continuar el dialogo, esto se muestra desde otra parte de la vista, en el caso de que el cliente iniciara la conversación también se declara que estado esta “active” pero sin la necesidad de comenzar la conversación con una plantilla.

Cuando una conversación está abierta, lo que significa que el usuario está dentro de la conversación, ya sea enviando un mensaje o esperando una respuesta, se marca como “open”. Esto permite mostrar toda la conversación en el centro de la pantalla para una interacción fluida.

Finalmente, cuando se cambia a otra conversación después de haber abierto una, esta pasa al estado “read”, lo que indica que la conversación ya ha sido respondida o que su contenido ha sido leído.

Capítulo 5. Descripción funcional de la aplicación

Para poder probar el funcionamiento de la API se ha creado una interfaz gráfica sencilla con el objetivo de mostrar las funcionalidades que se ofrecen, para empezar la aplicación dispone de un inicio de sesión donde se podrá acceder con un usuario para ver la aplicación, una vez registrado, lo primero que se muestra es un panel de control, Figura 15, consta de un panel simple desde donde se puede acceder a las diferentes vistas de la aplicación para poder gestionar algunos activos.

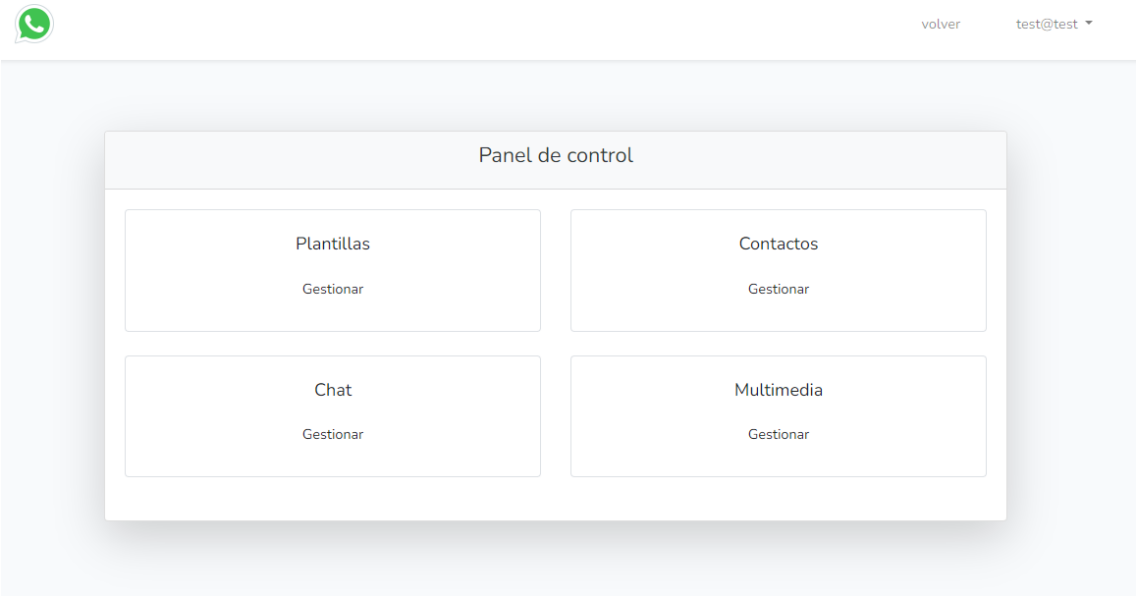


Figura 15 Panel de control de la aplicación

Comenzado por el apartado de plantillas, desde aquí se accede a una página donde se muestran las plantillas que se han añadido a la aplicación, desde esta pestaña que se muestra en la Figura 15, desde aquí se puede ver que tipos de plantillas esta cargadas en la aplicación para poder llevar una gestión de estas, además de poder realizar acciones sobre ellas como crearlas, editarlas, eliminarlas y finalmente enviarlas.

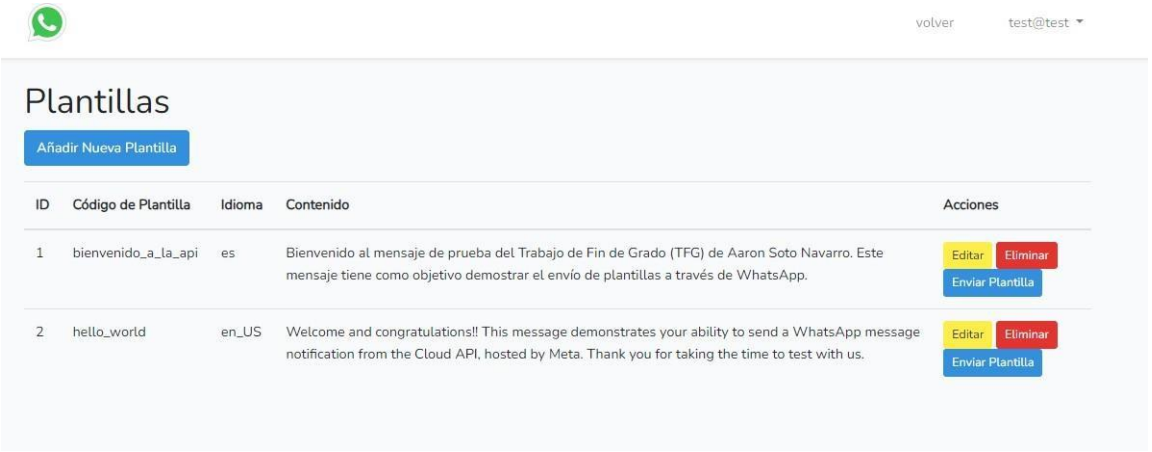


Figura 16 Índice vista plantillas

Para añadir una nueva plantilla desde la aplicación, se debe seleccionar el botón “Añadir nueva plantilla”, lo que redirige al usuario a la ventana ilustrada en la Figura 16. En esta, se deberá completar el formulario correspondiente, rellenando campos como el “código de plantilla”, que se refiere al título asignado a la plantilla, como se mostró en el apartado anterior de creación de plantilla, y el "idioma" en el que está configurada la misma. También se deben proporcionar

detalles sobre el “contenido” de la plantilla y el “estado” en que se encuentra según la plataforma de WhatsApp. Es importante señalar que estos dos últimos campos son meramente informativos y están destinados a ser visualizados dentro de la aplicación, otorgando al usuario una visión clara sobre el estatus y la naturaleza de las plantillas gestionadas.

Figura 17 Añadir plantilla a la aplicación

Tras añadir la plantilla, el usuario es redirigido a la vista de la Figura 15. Desde esta pantalla, se puede gestionar el envío de la plantilla mediante la selección del botón de envío asociado a ella. Al pulsar “Enviar plantilla”, el usuario es llevado a la ventana representada en la Figura 17, en la que deberá seleccionar el destinatario de la plantilla. Una vez que esta ha sido enviada correctamente, el usuario es nuevamente redirigido a la vista de la Figura 15, este es el funcionamiento principal del apartado de plantillas al que se accede desde el panel de control, Figura 14.

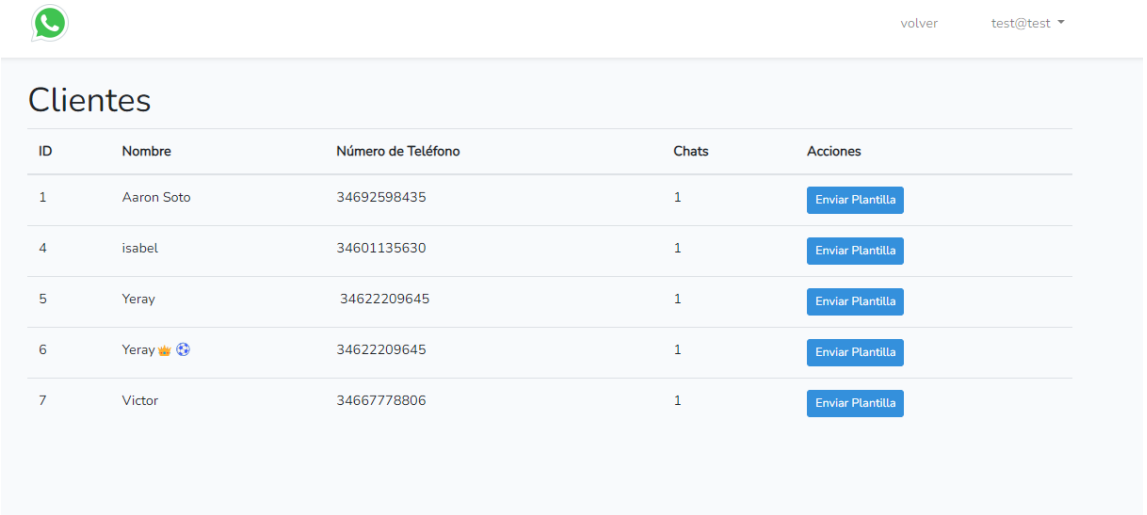


Figura 18 Vista envío de plantillas a clientes

Una vez vuelto al panel de control se encuentra el botón de multimedia, desde aquí se llega a una vista donde se muestran los archivos multimedia que se encuentran en la aplicación para luego poder ser enviado, Figura 18, desde el botón "Subir nuevo archivo multimedia", se accede a la página donde se van a subir los archivos multimedia a la plataforma de WhatsApp.

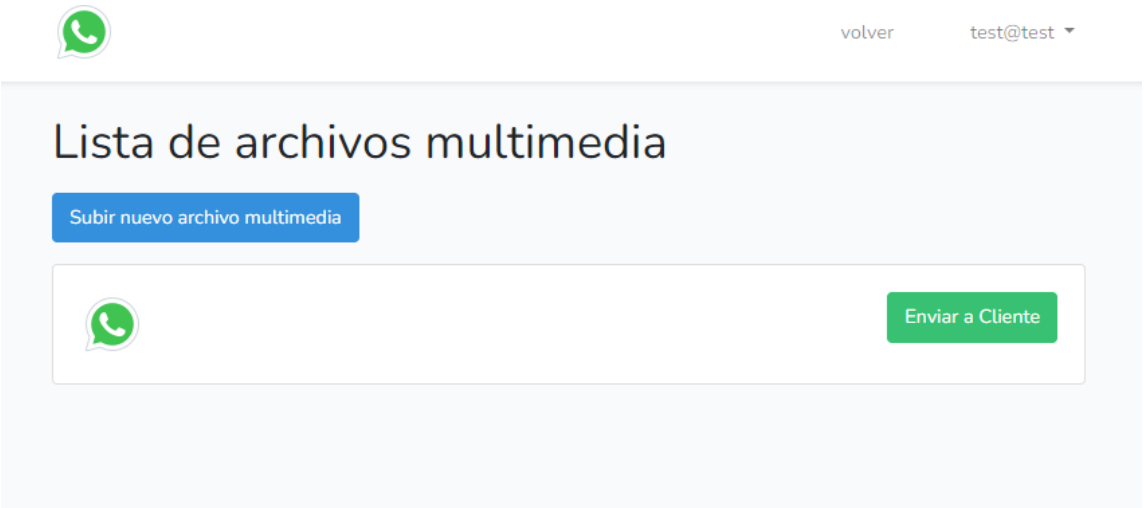


Figura 19 índice vista archivos multimedia

Desde la vista presentada en la Figura 19 se gestiona el proceso de obtención del identificador del archivo multimedia, el cual hay que subir previamente a la plataforma de WhatsApp para su posterior envío. En esta interfaz, los usuarios seleccionan un fichero que desean enviar, asegurándose de que cumpla con los tipos de archivos soportados y no exceda el límite de tamaño establecido por la plataforma. En caso de encontrar un error durante la subida, un mensaje se mostrará en esta misma vista, permitiendo realizar un nuevo intento. Si la subida se realiza de forma exitosa, el usuario será redirigido a la vista de la Figura 18.

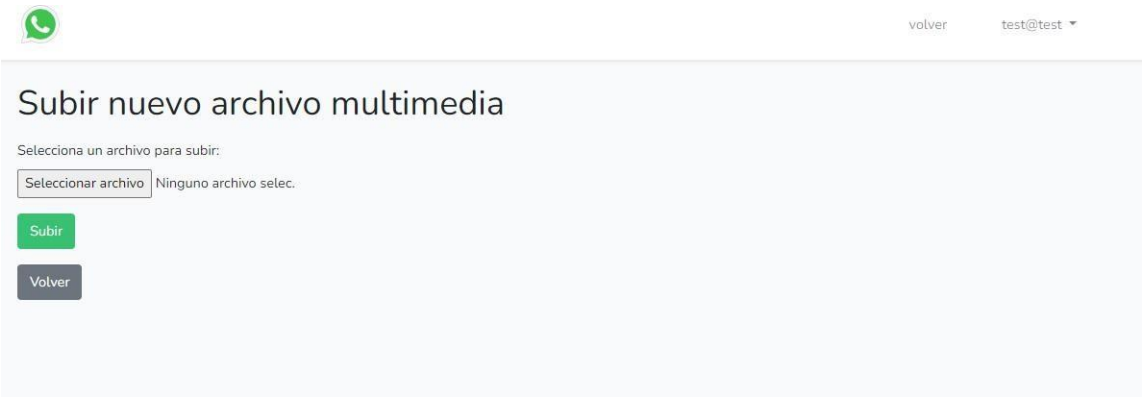


Figura 20 Vista subir archivos multimedia

Una vez que se han obtenido los identificadores para el envío de mensajes multimedia, desde la vista de la Figura 18, se puede especificar a qué cliente se desea enviar un mensaje. Este proceso es similar al envío de una plantilla: dirige al usuario a una página donde se muestran los clientes disponibles para el envío de este tipo de archivo. Al seleccionar uno, como se ilustra en la Figura

20, el mensaje se enviará al cliente elegido y, posteriormente, el usuario será redirigido a la vista donde se presentan todos los archivos multimedia disponibles.

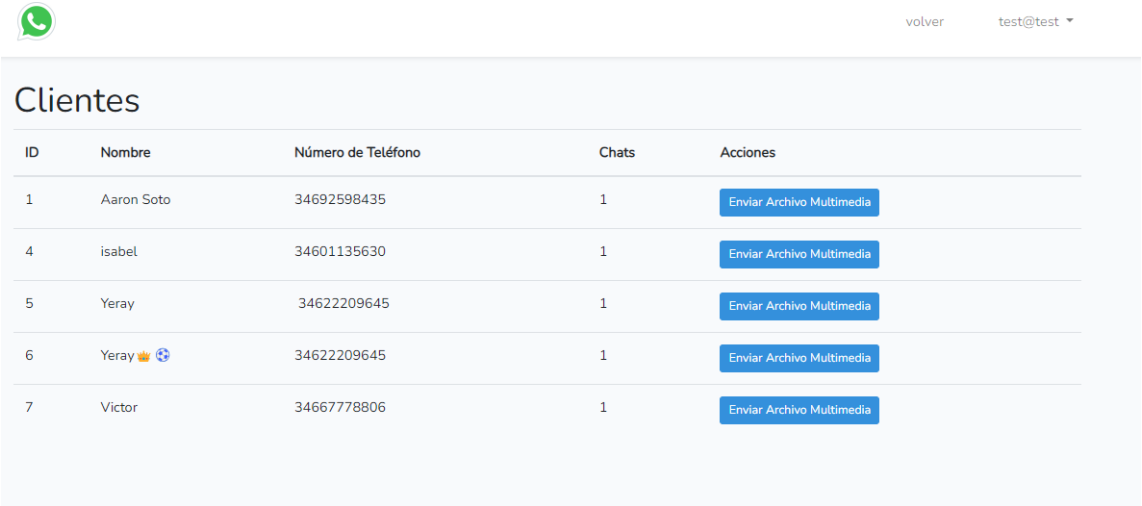


Figura 21 Vista de envío de mensajes multimedia a clientes

Después de explorar estas dos secciones del panel de control y si se desea gestionar los clientes de la aplicación a los que se quiere enviar mensajes, se encuentra disponible la sección de clientes. Al seleccionar esta opción desde el panel de control, se abrirá la página de clientes. Desde aquí, es posible visualizar a los clientes que, o bien han sido añadidos manualmente, o bien han iniciado una conversación directa con la aplicación, tal como se muestra en la Figura 21.

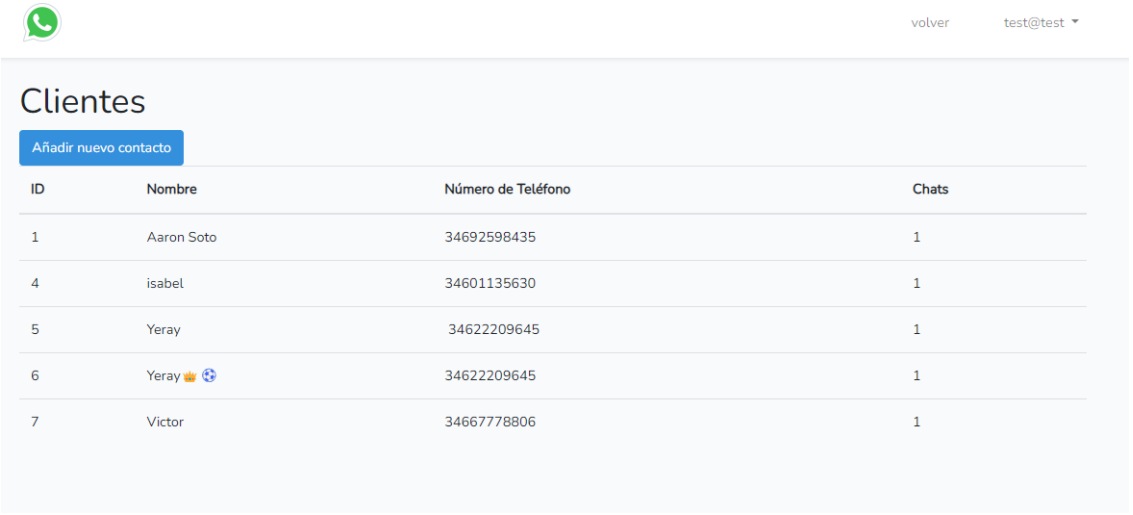


Figura 22 Vista clientes

Para enviar un mensaje a un nuevo cliente, primero es necesario añadirlo a la aplicación, proceso que implica el envío de un mensaje tipo plantilla para iniciar la conversación. Al seleccionar "Añadir nuevo contacto", el usuario será dirigido a la página de añadir clientes, representada en la Figura 22. En esta página, se deberán completar los campos pertinentes y seleccionar el mensaje tipo plantilla que se desea enviar. Tras enviarlo, el usuario será redirigido a la vista de clientes. Este paso es crucial para crear un chat que se asocie con el nuevo cliente y para enviar la plantilla que inicializará la comunicación.

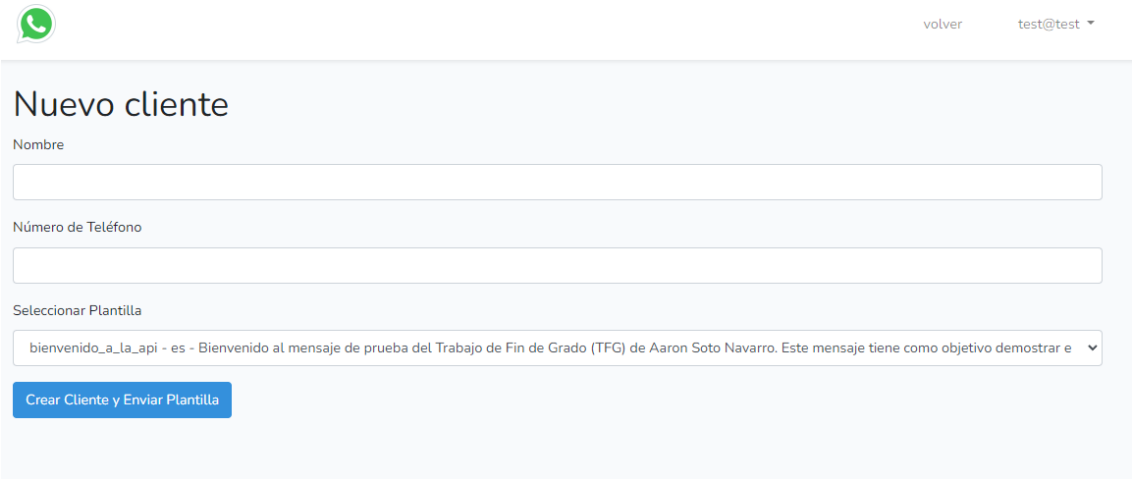
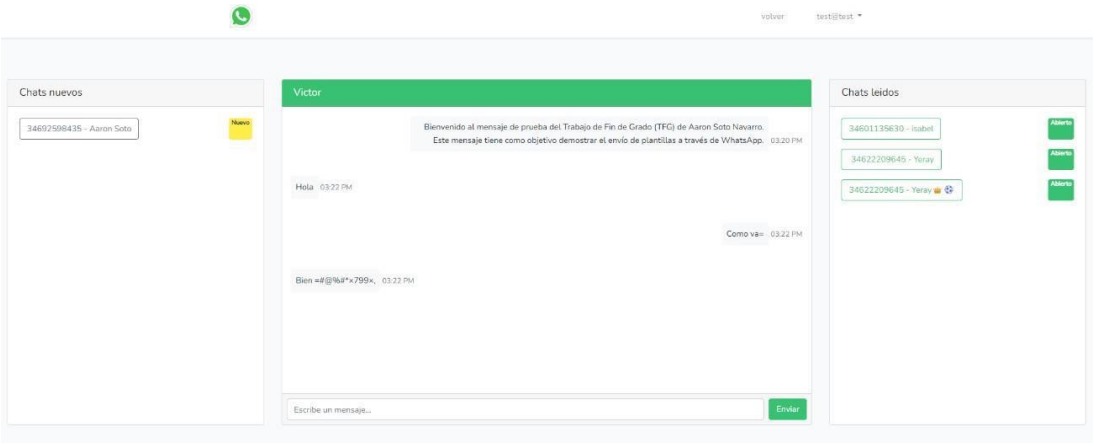


Figura 23 Vista añadir cliente

Finalmente, desde el panel de control podemos acceder a la sección de chat en tiempo real, la cual presenta diversas secciones. La primera parte consiste en un panel que muestra los chats nuevos recibidos por parte de los clientes. La sección central exhibe el chat que se encuentra abierto en ese momento. A la derecha, se visualizan las conversaciones que han sido respondidas, así como aquellas que se han iniciado mediante una plantilla, y que están a la espera de respuesta.



Capítulo 6. Conclusiones y propuestas de trabajo futuro.

6.1 Conclusiones

La exploración de la API de WhatsApp Business Cloud para ofrecer soporte técnico a través de la aplicación revela una serie de ventajas y limitaciones. A pesar de ser una alternativa aceptable y proporcionar un canal directo y accesible de comunicación con los usuarios, ciertas restricciones en la operativa y gestión de las conversaciones pueden obstaculizar una interacción cliente-empresa completamente fluida. Específicamente, la necesidad de utilizar plantillas para iniciar conversaciones después de un período de inactividad y la orientación de la API, principalmente diseñada para fines de marketing y ventas, pueden imponer restricciones en la adaptabilidad y espontaneidad de la comunicación cliente-empresa. No obstante, se ha observado que la API tiene un potencial considerable para integraciones con chatbots en campañas y estrategias comerciales.

6.2 Trabajo futuro

La API de WhatsApp Business Cloud, aunque presenta limitaciones, todavía alberga un significativo potencial para el desarrollo de futuras estrategias de soporte y ventas si se utiliza de manera ingeniosa y se integra con otras herramientas tecnológicas. Para trabajos futuros, es esencial explorar las posibilidades de integración con la nueva API de administración de WhatsApp Business, que proporciona un mayor control y gestionabilidad de los elementos previamente configurados manualmente a través de la plataforma de WhatsApp. Esto incluye la gestión de números de teléfono y la creación de plantillas, entre otros elementos.

Una propuesta sería desarrollar un sistema integrado que combine ambas APIs, de mensajería y administración, para crear una plataforma unificada de comunicación y gestión de soporte técnico. Este sistema podría incluir funcionalidades como la asignación automática de tickets de soporte, categorización de consultas mediante el uso de inteligencia artificial, y automatización de respuestas para consultas frecuentes. Asimismo, se podría explorar la implementación de interfaces de usuario más intuitivas y centros de ayuda automatizados para facilitar la autoayuda y, por ende, mejorar la experiencia del cliente y reducir la carga de trabajo del equipo de soporte.

También podría ser provechoso investigar cómo otras tecnologías, como la inteligencia artificial, el aprendizaje automático y las bases de datos en tiempo real, pueden ser integradas con la API para ofrecer soluciones más robustas y automatizadas que mejoren la interacción con el cliente, la gestión de datos y la efectividad de las campañas de marketing.

Además, desde el lado del cliente, se podrían emplear nuevas tecnologías para desarrollar una interfaz gráfica más intuitiva y estética, así como funcionalidades que permitan mostrar de una forma más adecuada algunas de las funciones en tiempo real. Laravel permite la integración de tecnologías como Angular o Vue.js, frameworks de JavaScript que facilitarían incorporar una estética y unas funcionalidades más agradables para el usuario.

Capítulo 7. Bibliografía

[1] WhatsApp, "Whatsapp.com," [En línea]. Disponible: https://www.whatsapp.com/legal/business-policy?lang=es_LA.

[2] Meta Platforms, Inc., "Categorización de plantillas," [En línea]. Disponible: https://developers.facebook.com/docs/whatsapp/updates-to-pricing/new-template-guidelines.

[3] Meta Platforms, Inc., "Números de teléfono," [En línea]. Disponible: https://developers.facebook.com/docs/whatsapp/cloud-api/reference/phone-numbers.

[4] Meta Platforms, Inc., "Límites de mensajes," [En línea]. Disponible: https://developers.facebook.com/docs/whatsapp/messaging-limits.

[5] Meta Platforms, Inc., "Precios basados en conversaciones," [En línea]. Disponible: https://developers.facebook.com/docs/whatsapp/pricing.

[6] Meta Platforms, Inc., "Información general," [En línea]. Disponible: https://developers.facebook.com/docs/whatsapp/cloud-api/overview.

[7] Laravel, "Laravel.com," [En línea]. Disponible: https://laravel.com/docs/7.x/.

[8] cre8ivelabs, "MVC in Web Development," [En línea]. Disponible: https://medium.com/@cre8ivelabs/mvc-in-web-development-2144849ef15a.

[9] M. &. T. J. Otto, "Getbootstrap.com," [En línea]. Disponible: https://getbootstrap.com/docs/5.3/getting-started/introduction/.

[10] JS Foundation-js. foundation., "JQuery," [En línea]. Disponible: https://jquery.com/.

[11] Mozilla.org, "JSON," [En línea]. Disponible: https://developer.mozilla.org/es/docs/Glossary/JSON.

[12] Pusher, "Puser Channels docs," [En línea]. Disponible: https://pusher.com/docs/channels/getting_started/javascript/?ref=docs-index.

[13] Microsoft, "Documentation for Visual Studio Code," [En línea]. Disponible: https://code.visualstudio.com/docs.

[14] Postman Inc., "What is postman?," [En línea]. Disponible: https://www.postman.com/product/what-is-postman/.

[15] Wikipedia contributors, "XAMPP," [En línea]. Disponible: https://es.wikipedia.org/w/index.php?title=XAMPP&oldid=153259874..

[16] A. Becker, "HeidiSQL - MariaDB, MySQL, MSSQL, PostgreSQL and SQLite made easy," [En línea]. Disponible: https://www.heidisql.com/.

[17] Nils Adermann y Jordi Boggiano, "Introduction - composer," [En línea]. Disponible: https://getcomposer.org/doc/00-intro.md.

[18] Wikipedia contributors, "GitHub," [En línea]. Disponible: https://es.wikipedia.org/w/index.php?title=GitHub&oldid=152403395..

[19] E. Plesky, "Plesk for developers. Online development platform," [En línea]. Disponible: https://www.plesk.com/developers/.

[20] Meta Platforms, Inc., "Webhooks Notification Payload Reference," [En línea]. Disponible: https://developers.facebook.com/docs/whatsapp/cloud-api/webhooks/components.